uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Electrónica Industrial y Automática
2016-2017

Trabajo Fin de Grado

"Sistema de captura de muestras biométricas para evaluaciones según ISO/IEC 19795"

Miguel Pintor Montes

Tutor

Raúl Sánchez Reíllo

Escuela Politécnica Superior, Leganés 16/10/2017

Dicen
que hace falta desorientarse,
una y otra vez,
para saber quién eres.

Y sobre todo, encontrar ese punto cardinal
donde todos soñamos con llegar.

Y allí,
aprender de tus errores,
asustar a tus miedos,
y disfrutar con tus locuras.

Agradecimientos

A mis padres, porque sin su apoyo y ayuda no habría sido posible tener una buena educación.

A Helga y Raúl, porque sin su colaboración inestimable no podría haber configurado las librerías.

Resumen

Este proyecto se centrará en la implementación de una aplicación informática para el reconocimiento automático de personas a través de sus patrones biométricos de huella dactilar, la cual se desarrollará en el lenguaje de programación C++.

En este proyecto habrá dos partes claramente diferenciadas:

- La primera de estas dos partes consiste en la gestión de los usuarios que tendrán acceso al sistema de reconocimiento de huella mediante una arquitectura cliente servidor. Los usuarios serán almacenados en una base de datos MySQL la cual será accedida mediante Scripts en PHP (nuestro servidor) que serán instanciados desde el código C++ (nuestro cliente). Las llamadas realizadas desde el cliente al servidor se realizarán mediante peticiones web, y el envío de datos va a realizarse mediante el método POST. Para llevar a cabo dichas peticiones web se hará uso de la librería cURL, que fue diseñada para este cometido.
- La segunda parte del proyecto será la implementación del sistema de reconocimiento de huella dactilar en sí, que estará formado en su fase inicial por el registro de los usuarios. Posteriormente el sistema recibirá un identificador de los usuarios y hará el reclutamiento de huella para cada uno de ellos. Finalmente se realizará la verificación de las huellas en dos fases distintas; La primera de estas fases tendrá la denominación de “primera visita” y la segunda fase será la “segunda visita”. Para todos estos procedimientos se utilizarán el hardware y las librerías que provee NITGEN, empresa fabricante de tecnología para detección biométrica. En concreto, como sensor de huella dactilar utilizaremos el modelo MFDU01-C3, y las librerías utilizadas para la implementación del algoritmo serán las facilitadas por el SDK eNBSP SDK.

Más adelante se pormenorizarán los pasos llevados a cabo para desarrollar del proyecto, explicando en detalle cada una de las distintas partes del mismo, así como haciendo un análisis de las distintas tecnologías empleadas para llevarlo a cabo.

Finalmente se expondrán tanto la planificación seguida como la estimación del presupuesto para su ejecución.

Abstract

This project is mainly focused on the implementation of a computer application for people automatic recognition through their biometric fingerprint patterns, which has been developed in C ++ programming language.

There will be two clearly differentiated parts within the project:

- The first of these two parts will consist on the management of the users who will have access to the fingerprint recognition system. This management will be developed using client-server architecture. Users will be stored in a MySQL database which will be accessed through scripts in PHP (our server) that are called from the C ++ application (our client). Calls made from the client to the server will be made via web requests. Data will be send using POST method. Web requests will be made use cURL library, which was designed for this purpose.
- The second part of the project will be the detection of footprint itself. The first step of this part includes user's registration which also can be considered user's management. Afterwards, the system will receive the user's ID and fingerprint enrolment will be done. Finally, the verification of the fingerprints will be carried out in two different phases; the first of these phases will be called "first visit" and the second phase will be the "second visit". For all these procedures will be used the hardware and the libraries provided by NITGEN which is a manufacturer of technology for biometric detection. Specifically, as a fingerprint sensor we will use the model MFDU01-C3, and the libraries used for the implementation of the algorithm will be facilitated by the SDK eNBSP SDK.

The development of the project will be described later explaining in detail each of the different parts it has, as well as a bibliographic review and an analysis of all different technologies used to carry it out.

Finally, both initial planning and execution's estimated budget we will be explained.

Índice

Índice de Figuras

Índice de Tablas

Listado de Acrónimos

API	Application Programming Interface
CPU	Central Processing Unit
CSV	Comma Separated Values
DBA	Database Administrator
DBMS	Database Management System
DNI	Documento Nacional de Identidad
GPL/Licencia	Licencia de derechos de autor más ampliamente usada en el mundo del software libre y código abierto
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
PC	Personal Computer
PDA	Personal Digital Assistant
PHP	Hypertext Preprocessor
SDK	Software Development Kit
SGBD	Sistema de Gestión de Base de Datos
SQL	Structured Query Language
TFG	Trabajo Fin de Grado
URL	Uniform Resource Locator
XAMPP	X (para cualquiera de los diferentes sistemas operativos), Apache, MariaDB, PHP, Perl

1 Introducción

1.1 Motivación y objetivos

La seguridad es un aspecto que forma parte de la vida cotidiana. La necesidad de contar con un ambiente confiable se encuentra presente en cualquier situación de la vida donde exista un flujo de información personal. En la actualidad, existe una amplia variedad de sistemas basados en la determinación o confirmación de manera fiable y robusta de la identidad de personas. Es en este contexto donde surge el reconocimiento biométrico o biometría, que corresponde al reconocimiento de personas a partir de sus características fisiológicas y/o de comportamiento, siendo la biometría dactilar una de las más empleadas, debido a su característica única para cada persona y fácil acceso. A partir de los años 60 comenzaron a desarrollarse los primeros sistemas automáticos de identificación basados en técnicas biométricas. En principio estas técnicas de reconocimiento tenían gran interés porque suponían una forma segura y sencilla de identificación de personas; Esto hizo posible el abandono de los sistemas tradicionales que conllevaban riesgos de seguridad importantes (basados en tarjetas de identificación y/o contraseñas que podían ser perdidas, olvidadas o incluso sustraídas). Por estas razones, los sistemas biométricos han experimentado un enorme crecimiento y actualmente están presentes en multitud de escenarios: control de pasajeros en aeropuertos, autenticación de documentos, control de acceso a zonas restringidas, gestión y control de asistencia laboral en empresas, etc. Los sistemas de reconocimiento biométrico usan características fisiológicas o de comportamiento, propias de cada individuo para identificarlo, es decir, se reconoce al usuario por lo que es en lugar de por lo que tiene o sabe.

Para la aplicación de la biometría dactilar, hoy en día se necesita un sensor especializado que permita obtener la huella dactilar y a partir de esta, mediante un tratamiento informático de la información obtenida a través del sensor, extraer las características necesarias para la identificación de la persona. Adicionalmente, para evaluar el funcionamiento del sistema de reconocimiento de huella dactilar desarrollado se necesita hacer uso de algún tipo de base de datos.

A pesar de que la huella dactilar es el sistema biométrico más utilizado en la actualidad, uno de sus mayores problemas radica en la dificultad para la identificación de personas mayores o trabajadores manuales, por el deterioro de la piel.

Por otro lado, los sistemas biométricos de iris presentan grandes índices de precisión y fiabilidad; sin embargo, los dispositivos para la captura del iris son demasiado costosos. De este modo, un sistema biométrico basado en imágenes de la huella dactilar se convierte en una buena alternativa para aplicaciones comerciales, obteniendo un equilibrado balance entre rendimiento y facilidad de uso. Otra de las ventajas de la huella dactilar es la facilidad con la que pueden extraerse las principales características mediante imágenes de muy baja resolución, motivo por el cual los dispositivos de captura puedan resultar mucho más económicos.

El objetivo general de este TFG es estudiar, diseñar e implementar un sistema capaz de llevar a cabo los procesos de adquisición y análisis de huella dactilar mediante el uso de un sensor de huella óptico. Para la realización de las pruebas se hará un registro de usuarios en una base de datos. Este tipo de aplicaciones requieren de reducidos tiempos de respuesta que posibilitan la disminución la complejidad de los algoritmos en cada etapa del sistema. Un control de acceso, como por ejemplo la entrada a un recinto deportivo donde se necesite garantizar la seguridad y conocer con exactitud quien ha accedido, podría ser una aplicación de este sistema.

Por el momento, el objeto de este proyecto es una primera aproximación a un sistema de reconocimiento de huella dactilar, de manera que los resultados obtenidos podrán servir de base para futuras investigaciones y desarrollo de sistemas comerciales para usos específicos del reconocimiento de huella dactilar.

1.2 Entorno socio-económico y marco regulador

Debido tanto al alto grado de seguridad que ofrecen los sistemas de huella dactilar como a sus precios competitivos, este tipo de sistemas están muy extendidos en la actualidad. Sus principales usos son en materia de seguridad, y en control de accesos. Las principales inversiones las realizan las inversiones públicas, y un claro ejemplo que podemos observar en el desarrollo de esta tecnología es la identificación que se hace al renovar el DNI; antes el registro de huella se realizaba con tinta, ahora desde hace tiempo ya se realiza de forma digital.

La normalización en el ámbito de la biometría comienza a gestarse a finales de los años 90 del siglo XX, cuando se empieza a ver la necesidad de crear interfaces comunes, así como formatos de datos conocidos.

Sin embargo, a consecuencia de los eventos del 11 de septiembre de 2001, se empuja esa necesidad, y sobre todo, el hecho de que los acuerdos sean de índole mundial. Esto motiva que el 20 de agosto de 2002, se cree un Subcomité (SC37) dedicado a la Identificación biométrica, dentro del Comité Conjunto ISO/IEC sobre Tecnologías de la Información (JTC1), en el seno de la Organización Internacional de Normalización (ISO) y de la Comisión Electrotécnica Internacional (IEC).

Para llevar a cabo el trabajo de normalización, se estableció una estructura dividida en seis Grupos de Trabajo (WG – Working Groups), cada uno de ellos destinado a trabajar en un aspecto determinado del campo de la Identificación Biométrica:

WG1: encargado de crear un catálogo de términos estandarizados que cubran todos aquellos conceptos relacionados con los sistemas biométricos.

WG2: cuyo ámbito de trabajo incluye los interfaces de comunicación entre aplicaciones y sistemas.

WG3: destinado a estipular los formatos de datos para cada una de las distintas modalidades biométricas.

WG4: cuya misión es la de definir las distintas arquitecturas biométricas y los distintos perfiles de aplicación.

WG5: encargado de definir los mecanismos de evaluación de los sistemas biométricos, desde el punto de vista de la funcionalidad, así como de indicar la forma en la que deben hacerse los informes de dichas evaluaciones.

WG6: que se encarga de estudiar los efectos jurídicos de la instalación de sistemas biométricos, así como la influencia social de su aplicación.

Durante los primeros años de vida, el Subcomité ISO/IEC JTC1/SC37, se ha caracterizado por un intenso y fructífero trabajo. Ya cuenta con cinco docenas largas de estándares publicados, otros en fases finales de elaboración, y muchas vías de trabajo abiertas.

Dentro de los estándares más importantes se encuentra la familia ISO/IEC 19794 sobre los formatos de datos de las modalidades biométricas, la ISO/IEC 19795 sobre la Metodología de Evaluación, y la ISO/IEC 19784 sobre Interfaces de Programación (BioAPI).

El desarrollo de la aplicación se hará siguiendo el estándar ISO/IEC 19795.

La norma ISO/IEC 19795-1 se refiere a la evaluación de sistemas biométricos en términos de tasas de error y tasas de rendimiento. Las medidas para las diversas tasas de error en el registro biométrico, la verificación y la identificación se especifican sin ambigüedad. Se ofrecen recomendaciones y requisitos para la realización de evaluaciones de desempeño a través de las etapas de planificación de la evaluación; recopilación de datos de transacciones de inscripción, verificación o identificación; análisis de tasas de error; y la presentación de informes de los resultados. Los principios presentados son genéricos para todas las modalidades biométricas, aplicaciones y propósitos de prueba, y a ambas metodologías de pruebas offline y online. Estos principios ayudan a evitar el sesgo debido a la recolección inapropiada de datos o procedimientos analíticos, dan mejores estimaciones del rendimiento y aclaran los límites de aplicabilidad de los resultados de la prueba.

1.3 Estructura de la memoria

Este documento está estructura en 7 capítulos, cuyos contenidos se resumirán a continuación:

Capítulo 1: Introducción

Este capítulo sirve a modo introductorio de la memoria y explica la motivación que ha llevado al desarrollo de este proyecto y algunos objetivos que se pretenden llevar a cabo con su implementación. También trata sobre el marco regulador en el que están enmarcadas las tecnologías de detección biométrica en general, y de detección de huella dactilar en particular.

Capítulo 2: Estado del arte

En este capítulo se hace una revisión bibliográfica de todas las tecnologías utilizadas para el desarrollo del proyecto, desde la detección biométrica y de huella digital, hasta los lenguajes de programación y bases de datos empleadas para la arquitectura de la aplicación.

Capítulo 3: Plataforma de desarrollo

Este capítulo aborda los distintos elementos que conforman la plataforma en la que será desarrollado el proyecto. Pormenoriza las características de los distintos elementos (IDE, editor de texto, servidor web, gestor y editor de base de datos...) para así entender la elección de cada uno de ellos.

Capítulo 4: Diseño de la solución

Esta es la parte de la memoria en la que se describirá el diseño de todas las partes de nuestro proyecto. En algunos casos el diseño inicial resultó erróneo, con lo cual se explicará por qué lo fue y se detallará el diseño final llevado a cabo.

Capítulo 5: Desarrollo

Este capítulo servirá para describir los procesos técnicos llevados a cabo para convertir en algo real el diseño realizado para la aplicación.

Capítulo 6: Pruebas

En este capítulo se hablará de las pruebas y depuraciones realizadas sobre el proyecto para verificar su correcto funcionamiento. En el caso de detectar fallos se comentarán las acciones realizadas para corregir dichos fallos.

Capítulo 7: Conclusiones y líneas futuras

Se analizará si los propósitos iniciales del proyecto han sido exitosos y además se propondrán posibles ampliaciones o usos futuros del mismo.

2 Estado del arte

Este capítulo está dedicado a hacer una revisión bibliográfica de todas las tecnologías empleadas en el desarrollo del proyecto, empezando por lo general para terminar en lo particular en los casos que corresponda. Se empezará realizando una introducción de bases de datos para después profundizar más en la que está basado este proyecto, MySQL.

Posteriormente se hará una breve exposición de los dos lenguajes de programación empleados en el desarrollo.

Para finalizar, y como en la primera parte del capítulo se empezará explicando aspectos generales para después hablar sobre el proyecto. Primero se hablará sobre biometría en todos sus campos de investigación y desarrollo, para posteriormente centrarnos en el objeto de este proyecto que es la detección de huella dactilar.

2.1 Bases de datos

Para empezar, se explicará lo que es una base de datos; El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California, USA. Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido; una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Actualmente, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital, siendo este un componente electrónico, por tanto se ha desarrollado y se ofrece un amplio rango de soluciones al problema del almacenamiento de datos. Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos. Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

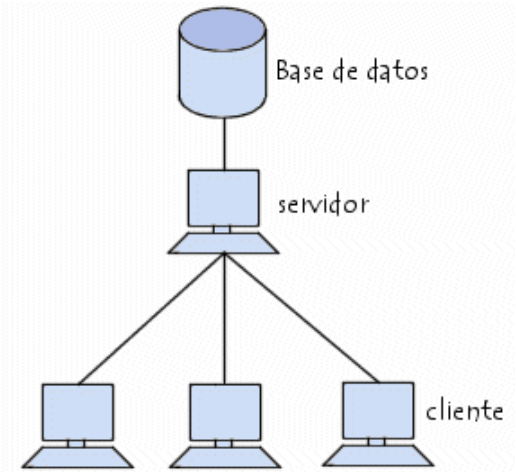


Fig. 1: Esquema bases de datos, servidores y clientes [1]

Una base de datos proporciona a los usuarios el acceso a datos, que pueden visualizar, ingresar o actualizar, en concordancia con los derechos de acceso que se les hayan otorgado. Se convierte más útil a medida que la cantidad de datos almacenados crece. Una base de datos puede ser local, es decir que puede utilizarla solo un usuario en un equipo, o puede ser distribuida, es decir que la información se almacena en equipos remotos y se puede acceder a ella a través de una red.
La principal ventaja de utilizar bases de datos es que múltiples usuarios pueden acceder a ellas al mismo tiempo.

Como componentes de una base de datos podríamos citar los siguientes:

- *Hardware:* constituido por dispositivo de almacenamiento como discos, tambores, cintas, etc.
- *Software:* que es el DBMS o Sistema Administrador de Base de Datos.
- *Datos:* los cuales están almacenados de acuerdo a la estructura externa y van a ser procesados para convertirse en información.

Rápidamente surgió la necesidad de contar con un sistema de administración para controlar tanto los datos como los usuarios. Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y *DataBase Management System*, su expresión inglesa.

Subsistema de un DBMS

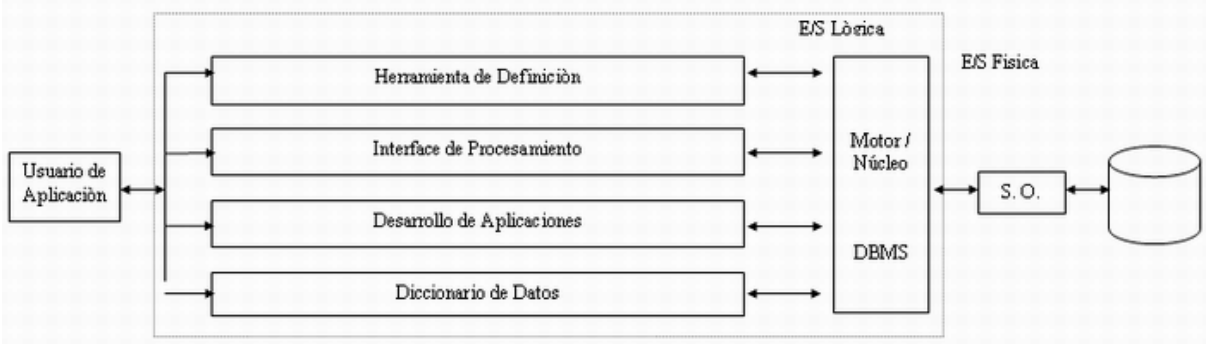


Fig. 2: Subsistema de un DBMS [2]

Motor ò Núcleo DBMS: recibe los requerimientos lógicos de E/S y los convierte en operaciones de lectura y escritura.
Lógicos: son cualquier tipo de consulta requerimiento de lectura con ingreso de datos (requerimiento de estructura) es ayudado por el Sistema Operativo para convertir estos requerimientos lógicos en físicos que actúan sobre dispositivos de almacenamiento.
Herramientas de definición: permite definir y modificar la estructura de la Base de Datos, a este nivel definimos lo que se conoce como "Esquema " que es la definición total de Base de Datos, es que definimos la estructura de la tabla, los tipos de campos, las restricciones para los campos.
- *Subesquema:* manejo de vistas de datos, de niveles externos.

- *Esquema:* manejo de niveles conceptuales.

Interfaz de Procesamiento: provee de las facilidades de actualización, despliegue y visualización de datos.
Desarrollo de Aplicaciones: permite generar una aplicación por Ej.: generadores de formas, pantalla, código, herramientas, case, etc.
Diccionario de Datos: este es el componente al subsistema con el que interactúan directamente los DBA, le proporciona niveles de consulta y reportes útiles para su trabajo de administración. Es la descripción de la estructura de Base de Datos y relaciones entre datos, y programas.

Ventajas de las bases de datos

Control sobre la redundancia de datos:

Los sistemas de ficheros almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además de provocar la falta de consistencia de datos. En los sistemas de bases de datos todos estos ficheros están integrados, por lo que no se almacenan varias copias de los mismos datos. Sin embargo, en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos.

Consistencia de datos:

Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantienen consistentes.

Compartir datos:

En los sistemas de ficheros, los ficheros pertenecen a las personas o a los departamentos que los utilizan. Pero en los sistemas de bases de datos, la base de datos pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados.

Mantenimiento de estándares:

Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.

Mejora en la integridad de datos:

La integridad de la base de datos se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

Mejora en la seguridad:

La seguridad de la base de datos es la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros.

Mejora en la accesibilidad a los datos:

Muchos SGBD proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.

Mejora en la productividad:

El SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros. A nivel básico, el SGBD proporciona todas las rutinas de manejo de ficheros típicas de los programas de aplicación. El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel.

Mejora en el mantenimiento:

En los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de aplicación que los manejan.

Esto hace que los programas sean dependientes de los datos, de modo que un cambio en su estructura, o un cambio en el modo en que se almacena en disco, requiere cambios importantes en los programas cuyos datos se ven afectados.

Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos, gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

Aumento de la concurrencia:

En algunos sistemas de ficheros, si hay varios usuarios que pueden acceder simultáneamente a un mismo fichero, es posible que el acceso interfiera entre ellos de modo que se pierda información o se pierda la integridad. La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y garantizan que no ocurran problemas de este tipo.

Mejora en los servicios de copias de seguridad:

Muchos sistemas de ficheros dejan que sea el usuario quien proporcione las medidas necesarias para proteger los datos ante fallos en el sistema o en las aplicaciones. Los usuarios tienen que hacer copias de seguridad cada día, y si se produce algún fallo, utilizar estas copias para restaurarlos.

En este caso, todo el trabajo realizado sobre los datos desde que se hizo la última copia de seguridad se pierde y se tiene que volver a realizar. Sin embargo, los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo perdido cuando se produce un fallo.

Desventajas de las bases de datos

Complejidad:

Los SGBD son conjuntos de programas que pueden llegar a ser complejos con una gran funcionalidad. Es preciso comprender muy bien esta funcionalidad para poder realizar un buen uso de ellos.

Coste del equipamiento adicional:

Tanto el SGBD, como la propia base de datos, pueden hacer que sea necesario adquirir más espacio de almacenamiento. Además, para alcanzar las prestaciones deseadas, es posible que sea necesario adquirir una máquina más grande o una máquina que se dedique solamente al SGBD. Todo esto hará que la implantación de un sistema de bases de datos sea más cara.

Vulnerable a los fallos:

El hecho de que todo esté centralizado en el SGBD hace que el sistema sea más vulnerable ante los fallos que puedan producirse. Es por ello que deben tenerse copias de seguridad (Backup).

Tipos de Base de Datos

Entre los diferentes tipos de base de datos, podemos encontrar los siguientes:

MySql: es una base de datos con licencia GPL basada en un servidor. Se caracteriza por su rapidez. No es recomendable usar para grandes volúmenes de datos.

PostgreSql y Oracle: Son sistemas de base de datos poderosos. Administra muy bien grandes cantidades de datos, y suelen ser utilizadas en intranets y sistemas de gran calibre.

Access: Es una base de datos desarrollada por Microsoft. Esta base de datos, debe ser creada bajo el programa Access, el cual crea un archivo .mdb con la estructura ya explicada.

Microsoft SQL Server: es una base de datos más potente que Access desarrollada por Microsoft. Se utiliza para manejar grandes volúmenes de informaciones.

2.2 MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.
Para que se entienda bien el concepto se pondrá un ejemplo. Cuando se tiene, por ejemplo, una página web, lo que se tiene es un código que al abrir la URL de la página web el navegador lee y lo convierte en algo visual y entendible. Este código se ayuda en el caso de las páginas web de los estilos CSS para darle una parte visual, de la programación PHP por ejemplo para generar acciones y procesar un contenido que tiene que estar almacenado de una forma.
MySQL es Open Source:
Open Source significa que la persona que quiera puede usar y modificar MySQL. Cualquiera puede descargar el software de MySQL de Internet y usarlo sin pagar por ello. Inclusive, cualquiera que lo necesite puede estudiar el código fuente y cambiarlo de acuerdo a sus necesidades. MySQL usa la licencia GPL (Licencia Pública General GNU), para definir qué es lo que se puede y no se puede hacer con el software para diferentes situaciones. Sin embargo, si uno está incómodo con la licencia GPL o tiene la necesidad de incorporar código de MySQL en una aplicación comercial es posible comprar una versión de MySQL con una licencia comercial. Para mayor información, ver la página oficial de MySQL en la cual se proporciona mayor información acerca de los tipos de licencias.
El servidor de bases de datos MySQL es muy rápido, seguro, y fácil de usar. Si eso es lo que se está buscando, se le debe dar una oportunidad a MySQL. Se pueden encontrar comparaciones de desempeño con algunos otros gestores de bases de datos en la página de MySQL.
El servidor MySQL fue desarrollado originalmente para manejar grandes bases de datos mucho más rápido que las soluciones existentes y ha estado siendo usado exitosamente en ambientes de producción sumamente exigentes por varios años. Aunque se encuentra en

desarrollo constante, el servidor MySQL ofrece hoy un conjunto rico y útil de funciones. Su conectividad, velocidad, y seguridad hacen de MySQL un servidor bastante apropiado para acceder a bases de datos en Internet.
MySQL está desarrollada en su mayor parte en ANSI C y C++. Es usada por muchos sitios web grandes y populares tales como Wikipedia, Google (aunque no la usa para sus búsquedas), Facebook, Twitter, Flickr, YouTube etc.
Existen varias interfaces de programación de aplicaciones que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL, incluyendo C, C++, C#, Pascal, Delphi (vía dbExpress), Eiffel, Smalltalk, Java (con una implementación nativa del driver de Java), Lisp, Perl, PHP, Python, Ruby, Gambas, REALbasic (Mac y Linux), (x)Harbour (Eagle1), FreeBASIC, y Tcl; cada uno de estos utiliza una interfaz de programación de aplicaciones específica. También existe una interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL. También se puede acceder desde el sistema SAP, lenguaje ABAP.
Algunos detalles técnicos de MySQL:
El software de bases de datos MySQL consiste de un sistema cliente/servidor que se compone de un servidor SQL multihilo, varios programas clientes y bibliotecas, herramientas administrativas, y una gran variedad de interfaces de programación (APIs). Se puede obtener también como una biblioteca multihilo que se puede enlazar dentro de otras aplicaciones para obtener un producto más pequeño, más rápido, y más fácil de manejar.

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad.

Poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferentes velocidades de operación, soporte físico, capacidad, distribución geográfica, transacciones...
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

Sentencias básicas de MYSQL

Como cualquier lenguaje para poder procesarse las acciones tienes que tener unas llamadas que devuelvan unos resultados. En MYSQL existen unas sentencias básicas que deberías

conocer para trabajar con este gestor de bases de datos: Conectarse por consola a una base de datos mysql: mysql -u usuario -p Mostrar todas las bases de datos:

SHOW DATABASES;

Usar una base de datos: Con esto nos referimos a seleccionar la base de datos con la que vas a hacer las acciones, antes de realizar una acción sobre una base de datos tienes que elegirla:

USE nombre_bd;

Mostrar tablas de la base de datos: Cada base de datos está estructurada en tablas, que agrupan la información de forma ordenada. Sería por hacer un símil como un bloque de casas que está organizada en plantas.

SHOW TABLES;

Mostrar todos los campos de una tabla: De igual forma que cada piso tiene habitaciones, las tablas de la base de datos tienen campos.

SELECT * FROM nombre_tabla;

Aquí podríamos hacer diferentes filtrados de búsqueda, como ordenar los resultados de una forma determinada, por un campo, orden alfabético… también podríamos buscar por un campo… la opciones son muchas. Crear una base de datos:

CREATE DATABASE nombre_bd;

Crear una base de datos:

DROP DATABASE nombre_bd;

Renombrar Base de datos:

RENAME TABLE nombre_bd1 to nombre_bd2;

…

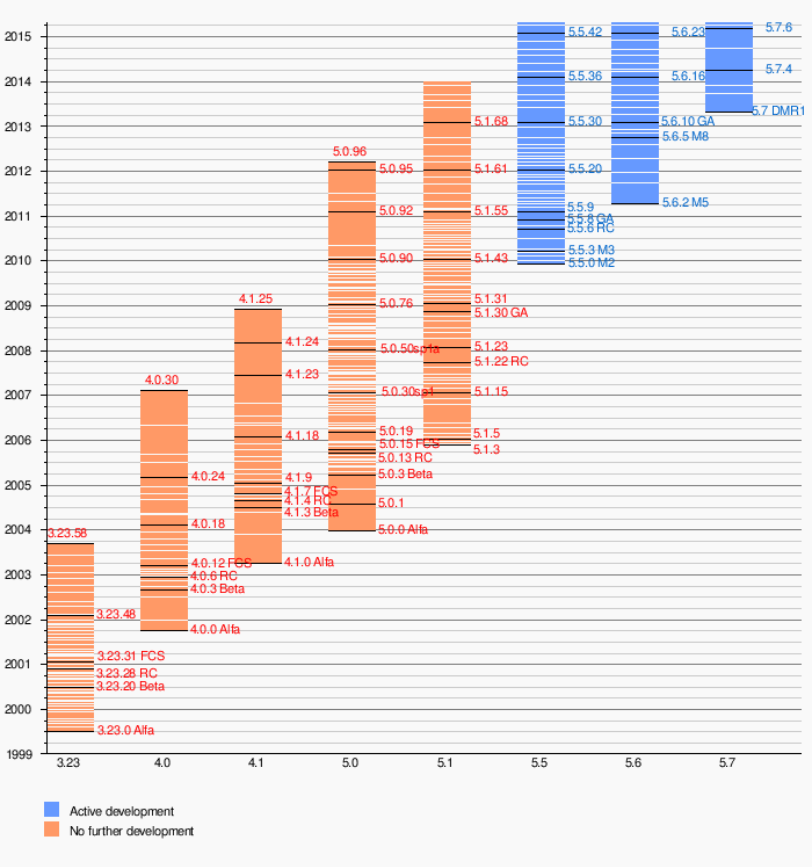


Fig. 3: Versiones de MySQL [3]

Según las cifras del fabricante, existirían más de seis millones de copias de MySQL funcionando en la actualidad, lo que supera la base instalada de cualquier otra herramienta de bases de datos. El tráfico del sitio web de MySQL AB superó en 2004 al del sitio de IBM.

2.3 Lenguaje C++

C++ es un lenguaje de programación diseñado a mediados de los años 80 por Bjarne Stroustrup. La intención de su creación fue el extender el exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, C++ es un lenguaje hibrido. Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y programación orientada a objetos). Por esto se suele decir que C++ es un lenguaje de programación multiparadigma. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos.

Una particularidad del C++ es la posibilidad de redefinir los operadores, y de poder crear nuevos tipos que se comporten como tipos fundamentales.

El nombre C++ fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre C con clases. La expresión C++ significa incremento de C y se refiere a que C++ es una extensión de C.

Los objetos en C++ son abstraídos mediante una clase. Según el paradigma de la programación orientada a objetos un objeto consta de:

- Identidad, que lo diferencia de otros objetos (Nombre que llevara la clase a la que pertenece dicho objeto).
- Métodos o funciones miembro.
- Atributos o variables miembro.

Un ejemplo de clase que podemos tomar es la clase perro. Cada perro comparte unas características (atributos). Su número de patas, el color de su pelaje o su tamaño son algunos de sus atributos. Las funciones que lo hagan ladrar, cambiar su comportamiento etc., serán las funciones de la clase.

El lenguaje C++ proporciona plantillas; Las plantillas son el mecanismo de C++ para implantar el paradigma de la programación genérica. Permiten que una clase o función trabaje con tipos de datos abstractos, especificándose más adelante cuales son los que se quieren usar. Por ejemplo, es posible construir un vector genérico que pueda contener cualquier tipo de estructura de datos. De esta forma se pueden declarar objetos de la clase de este vector que contengan enteros, flotantes, polígonos, figuras, fichas de personal, etc.

En C++ es posible definir clases abstractas. Una clase abstracta, o clase base abstracta (ABC), es una que está diseñada solo como clase padre de las cuales se deben derivar clases hijas. Una clase abstracta se usa para representar aquellas entidades o métodos que después se implementarán en las clases derivadas, pero la clase abstracta en sí no contiene ninguna implementación -- solamente representa los métodos que se deben implementar. Por ello, no es posible instanciar una clase abstracta, pero sí una clase concreta que implemente los métodos definidos en ella. Las clases abstractas son útiles para definir interfaces, es decir, un conjunto de métodos que definen el comportamiento de un módulo determinado. Estas definiciones pueden utilizarse sin tener en cuenta la implementación que se hará de ellos. En C++ los métodos de las clases abstractas se definen como funciones virtuales puras.

A pesar de su adopción generalizada, muchos programadores han criticado el lenguaje C ++, incluyendo Linus Torvalds, Richard Stallman, y Ken Thompson. Los problemas incluyen una falta de reflexión o recolector de basura, tiempos de compilación lentos, y mensajes de error detallados, particularmente de la metaprogramación de plantilla. Para evitar los problemas que existen en C ++, y para aumentar la productividad, algunas personas sugieren lenguajes alternativos más recientes que C ++, como D, Go, Rust o Vala.

2.4 Lenguaje PHP

PHP es un lenguaje de código abierto muy popular, adecuado para desarrollo web y que puede ser incrustado en HTML. Es popular porque un gran número de páginas y portales web están creadas con PHP.

Originalmente diseñado por el programador danés-canadiense Rasmus Lerdorf, en el año 1994 en base a la escritura de un grupo de CGI binarios escritos en el lenguaje C. En un comienzo, PHP sólo estaba compuesto por algunas macros que permitían trabajar más fácilmente en la creación de páginas web.

En el año de 1995 Rasmus Lerdorf le añadió el analizador sintáctico y se llamó PHP/F1 Versión 2, sólo reconocía texto HTML y algunas directivas de mSQL. Después de esta fecha la contribución al código fue pública.

PHP se utiliza para generar páginas web dinámicas. Recordar que llamamos página estática a aquella cuyos contenidos permanecen siempre igual, mientras que llamamos páginas dinámicas a aquellas cuyo contenido no es el mismo siempre. Por ejemplo, los contenidos pueden cambiar en base a los cambios que haya en una base de datos, de búsquedas o aportaciones de los usuarios, etc.

¿Cómo trabaja PHP? El lenguaje PHP se procesa en servidores, que son potentes ordenadores con un software y hardware especial. Cuando se escribe una dirección tipo http://www.aprenderaprogramar.com/index.php en un navegador web como Internet Explorer, Firefox o Chrome, ¿qué ocurre? Se envían los datos de la solicitud al servidor que los procesa, reúne los datos (por eso decimos que es un proceso dinámico) y el servidor lo que devuelve es una página HTML como si fuera estática. El esquema es: Petición de página web al servidor --> El servidor recibe la petición, reúne la información necesaria consultando a bases de datos o a otras páginas webs, otros servidores, etc. --> El servidor responde enviando una página web "normal" (estática) pero cuya creación ha sido dinámica (realizando procesos de modo que la página web devuelta no siempre es igual).

Fig. 4: Esquema funcionamiento PHP [4]

El lenguaje PHP presenta cuatro grandes características:

 Velocidad: PHP no solo es rápido al ser ejecutado sino que no genera retrasos en la máquina, por esto no requiere grandes recursos del sistema. PHP se integra muy bien junto a otras aplicaciones, especialmente bajo ambientes Unix.

 Estabilidad: PHP utiliza su propio sistema de administración de recursos y posee de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.

 Seguridad: PHP maneja distintos niveles de seguridad, estos pueden ser configurados desde el archivo .ini

 Simplicidad: Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente. Además PHP dispone de una amplia gama de librerías, y permite la posibilidad de agregarle extensiones. Esto le permite su aplicación en múltiples áreas, tales como encriptado, gráficos, XML y otras.

PHP se utiliza en millones de sitios; entre los más destacados se encuentran Wikipedia.org, Facebook.com y Wordpress.com.

2.5 Detección biométrica

La biometría es una rama de la biología que estudia y mide los datos de los seres vivientes. El término "biometría" deriva de la palabra griega "bios" (vida) y "metros" (medida) y en su significado corriente en informática, sirve para indicar la identificación automática o la verificación de la identidad de un sujeto, sobre la base de sus características físicas o de su comportamiento.

La "Biometría Informática" es la aplicación de técnicas biométricas a la autentificación e identificación automática de personas, en sistemas de seguridad informática. Las técnicas biométricas se basan en medir al usuario, directa o indirectamente, para reconocerlo automáticamente, aplicando técnicas estadísticas y de Inteligencia Artificial (lógica difusa, redes neuronales, etc.).

Un sistema biométrico común comprende cinco componentes:

 Un sensor utilizado para recopilar datos y convertir la información en formato digital.

 Algoritmos de procesamiento de señal que realizan actividades de control de calidad y desarrollan las plantillas biométricas.

 Un componente para almacenamiento de datos que contiene la información con la cual se comparan las nuevas plantillas biométricas.

 Un algoritmo de coincidencia que compara las nuevas plantillas biométricas con una o más de las plantillas almacenadas.

 Un proceso de decisión (ya sea automático o manual) que utiliza los resultados del componente de coincidencia para tomar una decisión basada en el sistema.

El reconocimiento biométrico de personas constituye un campo específico dentro de las áreas de procesado de señal y de reconocimiento de patrones, en el que se han realizado enormes avances y aportaciones científicas y tecnológicas en la última década. Esto ha conducido a que diversos organismos tecnológicos internacionales, entre los que destaca el NIST norteamericano (Nacional Institute of Standards and Technology), hayan considerado que las tecnologías de reconocimiento biométrico están suficientemente maduras para ser aplicadas a soluciones comerciales. Así, se han generado grandes expectativas en lo que respecta a la

implantación de este tipo de sistemas. No obstante, la progresión en la implantación industrial de sistemas biométricos parece estar por debajo de las previsiones realizadas hace pocos años. Este enfriamiento de las expectativas de implantación obedece a factores de muy diversa índole, entre los cuales podemos destacar:

- La fiabilidad de los sistemas, incluso los considerados de "alta seguridad" (como aquellos basados en huella dactilar o iris), que aún no ha alcanzado los niveles de funcionamiento esperados por la industria y los usuarios.
- La falta de realismo derivada de las tasas de error obtenidas en laboratorio (en forma de errores de falso rechazo y de falsa aceptación), tasas que se incrementan notablemente en las aplicaciones reales.
- La intrusividad de los sensores en determinadas modalidades, lo que obliga a los usuarios a ser necesariamente muy cooperativos.
- La vulnerabilidad de los sistemas frente a determinados ataques maliciosos.

Antecedentes y estado actual

El reconocimiento biométrico de personas supone una alternativa a los métodos de autenticación personal "clásicos", basados bien en "algo que el usuario conoce" (password, PIN, etc.), o bien en "algo que el usuario posee" (tarjeta, llave, etc.). A pesar de que los diversos rasgos biométricos son objeto de estudio desde hace varios decenios, especialmente desde los años 90, lo cierto es que su implantación en aplicaciones civiles no está tan extendida como cabría esperar. Los sistemas denominados clásicos siguen copando la mayor parte del mercado de autenticación personal, con una presencia testimonial de los sistemas de reconocimiento biométrico.

Inicialmente, los investigadores reaccionaron a dicha falta de implantación de sistemas biométricos centrándose fundamentalmente en la mejora de prestaciones y reducción de costes. Los niveles de precisión y prestaciones de sistemas así como el coste actual de los dispositivos han llevado a descartar las anteriores razones como causantes del problema; así, la comunidad tecnológica considera en la actualidad que la problemática causante de la falta de implantación de las tecnologías biométricas proviene de la baja fiabilidad de los sistemas de autenticación cuando se emplean en entornos operativos. Y se considera asimismo, que esto viene derivado –a su vez- de una falta de realismo de los experimentos realizados en laboratorio, que apenas tienen en cuenta aspectos tales como la seguridad, la privacidad, los fallos en la adquisición del rasgo, la deriva temporal del mismo, o la calidad de la señal adquirida por el sensor. Una primera solución aportada a las anteriores deficiencias consiste en el empleo de esquemas multimodales, mejorando así las prestaciones del sistema biométrico mediante la combinación de diferentes rasgos. Conviene en este punto resaltar que, de esta forma, resulta posible conciliar dos aspectos implicados: el empleo multimodal de modalidades consideradas menos fiables (comparadas, por ejemplo, con la huella o el iris) pero que resultan modalidades de alta transparencia y aceptabilidad, y bajo nivel de cooperatividad exigido al usuario. De esta forma, se tendrá un sistema global que incorporara la flexibilidad propia de éstas características de transparencia, aceptabilidad y cooperatividad, junto con resultados de autenticación multimodales comparables a aquellos que se tienen con modalidades de alta fiabilidad. Adicionalmente, estamos incrementando la flexibilidad del sistema, al permitir modalidades alternativas en el proceso de verificación/reconocimiento.

Como soluciones complementarias a lo anteriormente expuesto, y orientadas tanto al refuerzo de la robustez frente al fraude como a la mejora de la privacidad, se plantean: almacenamiento de datos biométricos seguro mediante el uso de técnicas de marcado al agua (watermarking), para evitar la reutilización de parámetros duplicados, la encriptación de las señales biométricas y de los modelos que éstas generan, y el almacenamiento y reconocimiento sobre tarjeta inteligente, conocido como match-on-card.

Algunos rasgos biométricos individuales

- Cara. Este rasgo es socialmente muy aceptado y suscita mucho interés y trabajo en el ámbito científico. Se han desarrollado multitud de técnicas para clasificación de caras. Generalmente las características en las que se basa el reconocimiento suelen obtenerse mediante descripciones espectrales localizadas (wavelets) o análisis de componentes principales (PCA), descomponiendo los espacios de test en conjuntos de eigenfaces. Otras alternativas incluyen el uso de modelos de grafos deformables, máquinas de vector soporte, modelos ocultos de Harkov (HMMs), o modelos de mezcla de Gaussianas (GMMs).

- Voz. La identificación de locutor es un tema en pleno desarrollo en el momento actual, aunque su grado de éxito depende fuertemente del escenario en que se enmarque la aplicación (p.ej. autenticación de grabación telefónica completa frente a comprobación de locución breve). Tradicionalmente las técnicas de reconocimiento se basan en la caracterización estadística de los parámetros homomórficos (dominio cepstral) que representan la envolvente espectral localizada de la voz del locutor, a nivel acústico. Actualmente se realizan importantes esfuerzos de investigación para la incorporación de rasgos de más alto nivel (suprasegmental, fonético, prosódico, idioléctico, léxico) como apoyo a la caracterización acústica.

- Mano. La geometría de la mano tiene un amplio uso en la práctica en control de inmigración, cárceles, etc. Estos sistemas generan vectores de características midiendo la distancia y ángulos entre dedos para generar un patrón o modelo de la mano. Asimismo existen desarrollos recientes haciendo uso de la textura propia de la palma de la mano (huella palmar). Se mejorará el estado del arte en relación con la adquisición de esta modalidad mediante el desarrollo de un sensor que no precise del apoyo y contacto directo de la mano.

- Firma. La firma manuscrita escrita es uno de los rasgos biométricos más aceptados personal, social y legalmente como medio de autenticación. Los métodos de reconocimiento tradicionales se basan en descripciones paramétricas globales de la firma y posterior mediada de distancia. El empleo de esquemas de modelado (como, por ejemplo, HMMs o GMMs) sobre las secuencia temporales derivadas de la adquisición on-line (velocidad, aceleración, ángulo de la tangente instantánea, presión) produce resultados competitivos. Desarrollos recientes se centran en la adaptación de este tipo de tecnologías a dispositivos de adquisición on-line tales como Tablets, PCs, PDAs, o teléfonos móviles de tercera generación.

Fig. 5: Rasgos biométricos estáticos [5]

Fig. 6: Rasgos biométricos dinámicos [6]

2.6 Detección de huella dactilar

Existen suficientes hallazgos arqueológicos que indican que las huellas dactilares se han venido utilizado en la identificación de individuos desde el año 6000 a.C. por diversas poblaciones asirlas y chinas [Jain 99a, Maltoni 03]. Entre ellos, cabe destacar, los restos de cerámica en arcilla con impresiones de huellas dactilares, que sugieren el empleo de las mismas como medio para identificar al alfarero. El primer estudio científico publicado sobre la estructura de crestas, valles y poros de las huellas dactilares data de 1684, realizado por el morfologista inglés Nehemiah Grew. Desde entonces, han sido muchos los investigadores que han trabajo en este campo. En 1823, por ejemplo, Purkinje propuso un esquema de clasificación de huellas en nueve clases atendiendo a la configuración de la estructura de crestas. En general, los estudios científicos de principios de 1800 llegaron a dos importantes conclusiones que, hasta hoy, han servido de base para el reconocimiento biométrico, especialmente en entornos forenses: la no existencia de dos huellas de individuos diferentes con un patrón de crestas coincidente, y la invariabilidad en el tiempo de dichos patrones

durante toda la vida. A principios de 1900, se admitieron las siguientes características biológicas de las huellas dactilares como base de la identificación de individuos:

- La estructura de crestas y valles de la epidermis de cada individuo es única y representa unívocamente su identidad.

- La estructura de crestas y valles de un individuo, aunque puede variar, lo hace dentro de unos límites tan reducidos, que hacen posible una clasificación sistemática.

- Los detalles de las estructuras de crestas y valles, así como las minucias, son particulares de cada individuo e invariables en el tiempo.

La primera y tercera característica constituyen los principios por los que se rige la identificación de individuos por sus huellas dactilares. La segunda característica constituye el principio que permite la clasificación de huellas dactilares.

La forma en la que se realiza el proceso de adquisición de huellas dactilares es muy diferente, dependiendo del tipo de aplicación biométrica en la que se van a procesar las imágenes obtenidas. El método tradicional de adquisición de huellas dactilares tintadas ha tenido siempre, y sigue teniendo lugar, en el ámbito judicial y forense. Actualmente, los sistemas automáticos de reconocimiento de huellas dactilares tienen también aplicación en otros contextos, como por ejemplo, los sistemas de acceso a entornos de seguridad, donde los requisitos de funcionamiento y exigencias de la aplicación, obligan al uso de técnicas on-line. En estos casos, las huellas son capturadas a través de dispositivos de adquisición electrónicos. El esquema general de estos dispositivos comprende las siguientes partes: *(i) sensor* de lectura, encargado de capturar la imagen de la huella; *(ii) conversar analógico/digital*, encargado de convertir la imagen analógica entregada por el sensor en una señal digital; *(iü) interfaz de comunicaciones* con otros dispositivos externos (por ejemplo, un ordenador personal). Dependiendo del principio físico de funcionamiento del sensor, se definen los diferentes tipos: óptico, de estado sólido y ultrasónico. En nuestra aplicación se utilizará un sensor óptico similar al de la figura [6.1].

Fig. 7: Lector de huella óptico [7]

Fig. 8: Lector de huella ultrasónico [8]

Características singulares de las huellas dactilares

 Delta. El punto delta, también conocido como punto singular extremo, se define como el punto más próximo al centro geométrico en el que tiene lugar la divergencia de las dos crestas de referencia. Puede tratarse de un punto perteneciente a un final de cresta, de un punto a partir del cual se produce una bifurcación de crestas, o de un punto perteneciente al valle en el que se produce la divergencia de las crestas de referencia.

 Núcleo. El núcleo o punto singular interno, se define como el punto situado sobre las crestas curvas más internas de la estructura. Debido a las diferentes estructuras de crestas curvas existentes, las reglas para seleccionar la posición del núcleo son muy complejas.

 Cómputo de crestas. Al igual que en el ámbito forense, un parámetro importante para establecer la clasificación automática de huellas dactilares es el cómputo de crestas entre puntos singulares. Es decir, el cómputo del número de crestas que cruzan la línea imaginaria que une una delta con un núcleo. Al igual que antes, debido a la complejidad en la configuración de las crestas, resulta difícil establecer un criterio para realizar el cómputo preciso de crestas.

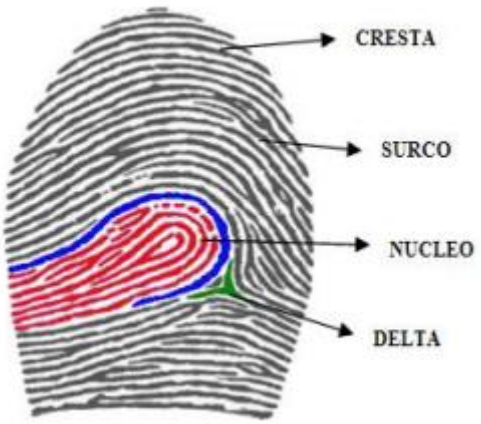


Fig. 9: Características de una huella [9]

Clases de huellas

Las diferentes clases o tipos de huellas entre las que debe diferenciar un clasificador, se establecen a partir de la información estructural de las crestas que determinan los puntos singulares. Existen en la literatura numerosos métodos y criterios de clasificación, dependiendo del ámbito de las aplicaciones. Una de las clasificaciones más aceptadas para los sistemas automáticos de reconocimiento establece la ordenación de las huellas dactilares en las siguientes categorías [Kawagoe 84, Srinivasan 92, Karu 96, Ratha 96, Cappelli 99]: arco sencillo, arco tensado, lazo derecho, lazo izquierdo, rizo y doble rizo.

- Lazo. Una huella dactilar de la clase lazo es aquella en la que las crestas entran por un lado de la imagen, fluyen hacia arriba rodeando a un núcleo con curvatura muy pronunciada, cortan la línea imaginaria que une el núcleo con una delta, y tienden a volver hacia el lado de la imagen por el que entraron. Aproximadamente, entre un 60% y un 65% de las huellas pertenecen a esta clase.

- Arco. El arco es una clase de huella muy particular, ya que a ella pertenecen menos del 5% de las huellas. Pueden darse dos tipos diferentes de arco: el arco sencillo y el arco tensado. En un arco sencillo las crestas fluyen de un lado al otro de la imagen, alzándose y curvándose ligeramente en el centro de la misma. En un arco tensado las crestas fluyen de un lado al otro de la imagen, como en un arco sencillo, pero curvándose de manera muy pronunciada al rodear a un núcleo.

- Rizo. Una huella pertenece a la clase rizo cuando la estructura de crestas presenta al menos dos puntos delta, frente a los cuales fluyen las crestas curvándose alrededor de un núcleo. Tal definición supone la existencia de un núcleo frente a cada delta. Las huellas del tipo rizo puede dividirse a su vez en: rizo sencillo y rizo doble. Aproximadamente, entre un 30% y un 35% de las huellas pertenecen a esta clase.

Fig. 10: Clases de huellas [10]

3 Plataforma de desarrollo

En este apartado se va a describir el entorno que se ha escogido para realizar la aplicación de detección de huella dactilar. Como podrá observarse de la lectura del anterior apartado, los lenguajes de programación que se han empleado han sido C++ y PHP.

3.1 IDE (Entorno de desarrollo integrado)

3.1.1 Visual Studio 2017

Fig. 11: Visual Studio 2017 [11]

Visual Studio es un conjunto de herramientas y otras tecnologías de desarrollo de software basado en componentes para crear aplicaciones eficaces y de alto rendimiento, permitiendo a los desarrolladores crear sitios y aplicaciones web, así como otros servicios web en cualquier entorno que soporte la plataforma.

En palabras más específicas, Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C# y Visual C++ utilizan todos el mismo entorno de desarrollo integrado (IDE), que habilita el uso compartido de herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes utilizan las funciones de .NET Framework, las cuales ofrecen acceso a tecnologías clave para simplificar el desarrollo de aplicaciones web ASP y Servicios Web XML.

3.1.2 Notepad ++

Fig. 12: Notepad++ [12]

Se ha utilizado Notepad++ para la escritura del código PHP.

Si hablamos de programas sencillos y ligeros que al mismo tiempo sean útiles para realizar múltiples tareas el Bloc de notas es uno de ellos. A pesar de ser increíblemente liviano nos sirve para muchas tareas, desde escribir un texto, hasta hacer una página web o, incluso un programa. Pero para usos más profesionales, el Bloc de notas se queda corto. Es ahí donde entra en juego el Notepad++.

Las características de Notepad++ son las siguientes:

 Coloreado y envoltura de sintaxis: si se escribe en un lenguaje de programación o marcado, Notepad++ es capaz de resaltar las expresiones propias de la sintaxis de ese lenguaje para facilitar su lectura.

 Pestañas: al igual que en muchos navegadores, se pueden abrir varios documentos y organizarlos en pestañas.

 Resaltado de paréntesis: cuando el usuario coloca el cursor en un paréntesis, Notepad++ resalta éste y el paréntesis correspondiente de cierre o apertura. También funciona con corchetes y llaves.

 Grabación y reproducción de macros.

 Soporte de extensiones: incluye algunas por defecto.

Notepad++ soporta una gran cantidad de lenguajes de programación.

3.2 Servidor web

Fig. 13: Servidor apache [13]

Apache: es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.11 y la noción de sitio virtual.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Las funcionalidades del servidor apache son las siguientes:

 Atender de manera eficiente, ya que puede recibir un gran número de peticiones HTTP, incluyendo una ejecución multitarea ya que pueden darse peticiones simultáneas. Cualquier petición compleja (por ejemplo con acceso a base de datos) dejaría colapsado el servicio.

 Restricciones de acceso a los ficheros que no se quieran ‘exponer’, gestión de autentificaciones de usuarios o filtrado de peticiones según el origen de éstas.

 Manejar los errores por páginas no encontradas, informando al visitante y/o redirigiendo a páginas predeterminadas.

 Gestión de la información a transmitir en función de su formato e informar adecuadamente al navegador que está solicitando dicho recurso.

 Gestión de logs, es decir almacenar las peticiones recibidas, errores que se han producido y en general toda aquella información que puede ser registrada y analizada posteriormente para obtener las estadísticas de acceso al sitio web.

3.3 Gestor de base de datos

Fig. 14: MySQL [14]

MySQL: es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Las características por las que utilizar este gestor de base de datos son las siguientes:

 Es Multiplataforma: Para S.O. como Windows, Linux y Mac disponemos de nuestro servidor para instalarlo.

 Es fácil encontrar ayuda: Al ser una base de datos que se utiliza en multitud de aplicaciones web existen multitud de tutoriales, foros, …. en la red en los que podemos encontrar la información que necesitamos

 Es fácil de aprender: Simplemente con conocer el estándar de SQL podemos manejar la base de datos MySQL si ningún problema.

 MySQL es una base de datos ampliamente probada por distintos usuarios y empresas con alto éxito.

 Menos características. Menos mantenimientos: Realmente esto nos da la ventaja para que un programador cualquiera pueda aprender rápidamente como debe mantener la base de datos para sus aplicaciones. Sin necesidad de ser un experto Administrador en Base de Datos (DBA). Bases de datos como Oracle requieren de DBA para la gestión de su información

debido a todas las características que tienes para su administración. En cambio MySQL para el funcionamiento habitual de una aplicación incluye unas características mínimas que nos sirven ampliamente para nuestras aplicaciones sin tener que recurrir a un DBA para que administre la base datos.

3.4 XAMPP

Fig. 15: XAMPP [15]

XAMPP es un paquete de instalación independiente de plataforma, software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MariaDB, PHP, Perl.

Seguridad

Oficialmente los diseñadores de XAMPP solo pretendían su uso como una herramienta de desarrollo, para permitir a los diseñadores de sitios webs y programadores testear su trabajo en sus propios ordenadores sin ningún acceso a Internet. En la práctica, sin embargo, XAMPP es utilizado actualmente como servidor de sitios Web, ya que, con algunas modificaciones, es generalmente lo suficientemente seguro para serlo. Con el paquete se incluye una herramienta especial para proteger fácilmente las partes más importantes en una página.

3.5 Editor de base de datos

Fig. 16: phpMyAdmin [16]

PhpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos. Se encuentra disponible bajo la licencia GPL.

Sus características son las siguientes:

 Interfaz Web para la gestión grafica

 Manejador de base de datos MySQL, MariaDB y Drizzle

 Importación de datos desde CSV y SQL

 Exporta datos a varios formatos: CSV, SQL, XML, PDF (vía la biblioteca TCPDF), ISO/IEC 26300 - OpenDocument Text y Spreadsheet, Word, Excel, LaTeX y otros

 Administración de múltiples servidores

 Crea gráficos PDF del diseño de la base de datos

 Crea consultas complejas usando Query-by-Example (QBE)

 Búsqueda global en una base de datos o un subconjunto de esta

 Transforma datos almacenados a cualquier formato usando un conjunto de funciones predefinidas, tal como BLOB

 Live charts para monitorizar las actividades del servidor MySQL tales como conexiones, procesos, uso de CPU/Memoria, etc.

3.6 eNBSP SDK

Fig. 17: Nitgen [17]

eNBSP SDK 4.0 es un kit de desarrollo de software que combina el SDK 3.0 existente (ahora denominado BSP - Biometric Solution Provider) y un algoritmo de reconocimiento de huellas 1:N no solo para aplicaciones básicas sino también para aplicaciones que utilizan las huellas de bases de datos de gran capacidad y donde se requiere una velocidad de búsqueda de huellas muy elevada. Este kit de desarrollo proporciona una interfaz de programación de alto nivel API (Application Programming Interface) que permite implementar un software con un interfaz de usuario de una forma fácil y rápida, ahorrando al programador tiempo y esfuerzos en el desarrollo de la aplicación. El kit de desarrollo permite operar en distintas plataformas puesto que soporta varios sistemas operativos y lenguajes de programación así como distintos dispositivos de reconocimiento de huella dentro de los productos fabricados con la tecnología de NITGEN. El kit de desarrollo ofrece unas condiciones óptimas para el desarrollo de soluciones de reconocimiento de huella mediante un sofisticado algoritmo de identificación que garantiza un alto grado de exactitud en el reconocimiento y una elevada velocidad de búsqueda de huellas.

4 Diseño de la solución

En este cuarto capítulo se explicará el diseño final llevado a cabo para la ejecución del proyecto, así como unas primeras consideraciones de diseño que finalmente fueron descartadas. En algunos casos se remarcarán aspectos comentados en el capítulo 3, ya que la elección de la plataforma de desarrollo está también incluida en las consideraciones de diseño.

4.1 Base de datos

4.1.1 Cometido de la base de datos

El primer paso en la aplicación es realizar una gestión de usuarios utilizando una base de datos. Para ello se creará en la aplicación un apartado de registrar usuarios y otro de buscar.

Para el registro de usuarios se tendrán en cuenta datos tales como el DNI, el nombre, apellidos, sexo, fecha de nacimiento, y también sus datos de contacto, que serán teléfono y correo electrónico. Una vez metidos los datos se tendrá que comprobar que el DNI es correcto mediante un algoritmo de comprobación de la letra, y además comprobar que el usuario no se encuentra ya dado de alta en la base de datos, comprobando el DNI, el teléfono y el correo. Si el usuario no se encuentra dado de alta previamente se le dará de alta en la base de datos.

Posteriormente, en el apartado de búsqueda se podrán buscar usuarios por DNI, correo electrónico, por teléfono, por nombre y por ambos apellidos. Devolverá un listado con todos los que hay que cumplan ese criterio, y después se elegirá uno y se mostrarán todos los datos de ese usuario.

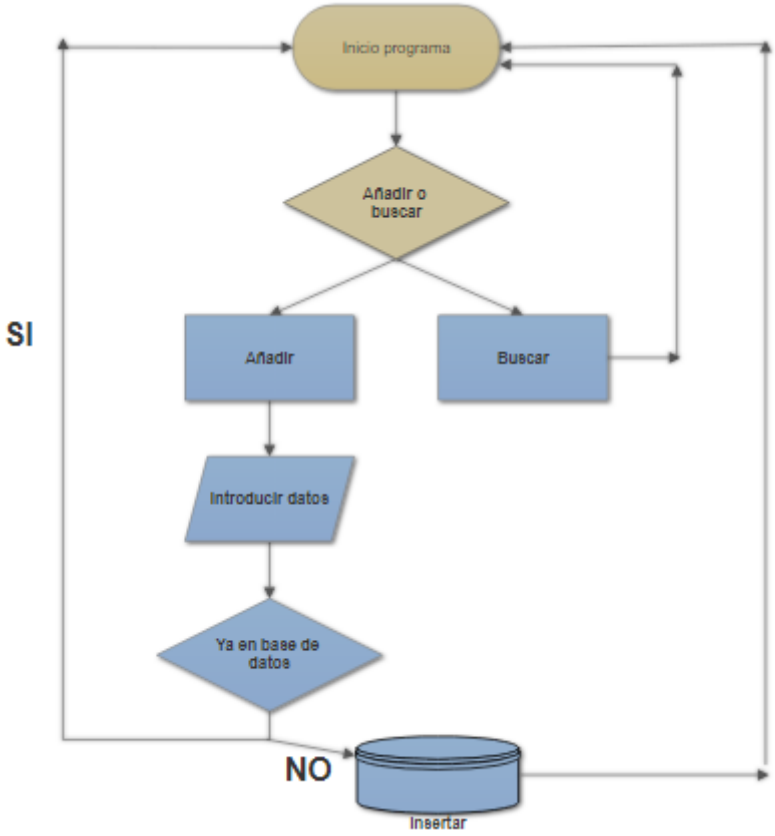


Fig. 18: Diagrama gestión usuarios

4.1.2 Diseño de la gestión de la base de datos

La base de datos no estará en local, sino en remoto, y como se comentó en el capítulo anterior, la arquitectura de base de datos elegida para hacer la gestión de usuarios ha sido MySQL ya que es la base de datos de código abierto más popular del mundo. Para ello se ha utilizado el servidor gratuito XAMPP ya que tiene integrado el servidor web Apache y la base de datos MySQL.

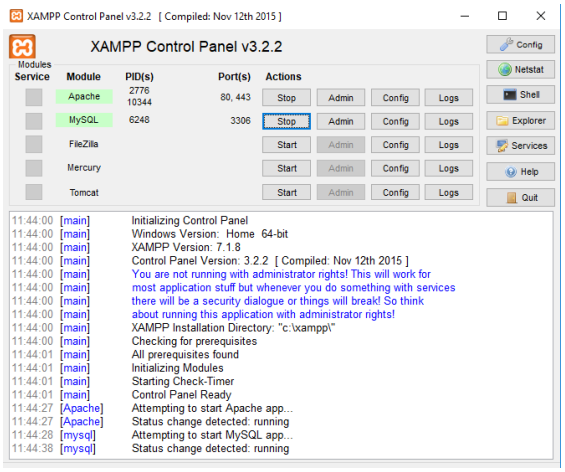


Fig. 19: XAMPP control panel

Ya que nuestra base de datos solo va a almacenar usuarios, solo será necesaria la creación de una tabla dentro de la misma. Para registrar usuarios, se guardarán los campos DNI, nombre, apellidos, sexo, fecha de nacimiento, teléfono y correo electrónico. Para almacenarlos se hará de la forma expuesta a continuación, usando los tipos de datos que facilita SQL:

 El DNI, nombre, apellidos y email se almacenarán en un array de caracteres. Tanto el DNI como el email serán claves únicas dentro de la base de datos, para posteriormente garantizar que no se incluya a ningún usuario dado de alta previamente.

 El sexo se codificará de forma binaria con un solo bit. Para realizarlo, MySQL ofrece el tipo de dato tinyint, lo cual es un booleano.

 Para almacenar la fecha de nacimiento, MySQL facilita el tipo de dato DATE, que almacena fechas según el formato AAAA-MM-DD.

 El teléfono se almacenará en un array de enteros, y al igual que DNI y email será de clave única por las mismas razones.

 Adicionalmente, se ha creado un identificador numérico para cada usuario, que se generará automáticamente cuando se produzca el alta. El dato será un entero auto incremental, y además de ser único, será la clave primaria de la base de datos.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	DNI	char(10)	latin1_swedish_ci		No	*Ninguna*		
2	NOMBRE	char(50)	latin1_swedish_ci		No	*Ninguna*		
3	APELLIDO1	char(50)	latin1_swedish_ci		No	*Ninguna*		
4	APELLIDO2	char(50)	latin1_swedish_ci		No	*Ninguna*		
5	SEXO	tinyint(1)			No	*Ninguna*		
6	FECHA_NAC	date			No	*Ninguna*		
7	TELEFONO	int(11)			No	*Ninguna*		
8	EMAIL	char(100)	latin1_swedish_ci		No	*Ninguna*		
9	ID	int(11)			No	*Ninguna*		AUTO_INCREMENT

Fig. 20: Estructura tabla usuarios

4.1.3 Acceso a la base de datos

El acceso a la base de datos ha sido uno de los aspectos clave en el diseño de la misma, y lo que más problemas ha dado. En un diseño inicial, se accedía directamente, ejecutando las sentencias propias del lenguaje SQL empotradas en la aplicación C++. Para ellos se iba a hacer uso de la librería "sqlca.h" Esta idea inicial se desechó para finalmente hacer un diseño mediante una arquitectura cliente-servidor. La arquitectura cliente-servidor es un modelo de diseño de software en el que tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

Las razones principales de acceder a la base de datos de esta forma son principalmente por motivos de seguridad y para evitar posibles puertas traseras. Para acceder a la base de datos se necesita un usuario y una contraseña, esta información viaja más segura con la arquitectura elegida finalmente.

El funcionamiento es el siguiente:

Nuestro cliente será la aplicación C++, y nuestro servidor serán distintos scripts programados en lenguaje php. El cliente comunicará con el servidor haciendo peticiones web y enviándole datos mediante el método POST. En este método la información no se envía por la URL sino que es invisible al usuario. Posteriormente el servidor hace una query a la base de datos, que le devuelve la información requerida en función de la query. El servidor genera un json con la información y le emite una respuesta al cliente, para que trate la información como sea conveniente.

El principal problema y dificultad ha sido realizar la petición web del cliente al servidor. De todos los métodos posibles, se ha hecho uso de la librería cURL para llevarla a cabo. La

siguiente imagen muestra un esquema de todo este proceso para un cliente Android, en nuestro caso el cliente es C++, pero el proceso sería el mismo.

Fig. 21: Esquema comunicación cliente-servidor [18]

4.2 Reconocimiento de huella

El fin de este proyecto es crear un programa que sea capaz de registrar huellas dactilares y posteriormente realice verificaciones. Para este fin, se ha hecho un diseño de la aplicación relativamente sencillo en tres fases que será pormenorizado más adelante.

4.2.1 Sensor

El sensor utilizado es el modelo MFDU01-C3, fabricado por SecuGen. El sensor está integrado dentro de un ratón de ordenador, de ahí la letra M del principio de su nombre; M de mouse y FDU01-C3 el modelo. La tecnología de reconocimiento de huellas dactilares SecuGen se basa en las minucias, que son los puntos únicos de una huella digital. Después de escanear una huella dactilar, se extraen las minucias y se procesan en una plantilla que se utilizará para la inscripción biométrica y emparejamiento. Las ventajas de los sensores SecuGen son las siguientes:

- **Excelente calidad de imagen:** Las imágenes de huellas dactilares se general mediante métodos ópticos y son claras y sin distorsión. Esto produce un mejor muestreo para la extracción de minucias.
- **Durabilidad:** Las pruebas de resistencia mecánica muestran una gran resistencia a los impactos, golpes y arañazos.
- **Software potente:** El algoritmo de procesamiento es preciso y rápido, lo que garantiza eficiencia y fiabilidad.

 Robustez y versatilidad: Sus materiales permiten el uso bajo condiciones extremas

 Diseño ergonómico: Tienen un diseño compacto y modular que permite una integración perfecta en dispositivos pequeños. Su facilidad de uso y compatibilidad hacen que los sensores SecuGen sean ideales para una amplia gama de aplicaciones.

 Low cost: Los productos son desarrollados para ofrecer alto rendimiento, y no necesitar mantenimiento a precios muy asequibles para el uso industrial.

Como curiosidad, añadir que el modelo en concreto que hemos utilizado ha sido muy cómodo ya que al estar integrado en el ratón y tener el lector de huella dactilar en el lugar de apoyo del dedo pulgar, ha sido muy conveniente su uso para realizar pruebas durante el desarrollo de la aplicación y no se ha requerido tener conectados varios periféricos al ordenador.

Fig. 22: Sensor MFDU01-C3 [19]

4.2.2 SDK

Como se mencionó en el capítulo anterior, para realizar la implementación de la aplicación de reconocimiento de huella dactilar se ha utilizado eNBSP SDK, en concreto la versión 4.8x para Windows. Este kit de desarrollo software soporta varios lenguajes de programación; C#, Visual Basic, Delphi… Por supuesto, también soporta C++ que es lo que nos interesa para nuestra aplicación. Las principales características del SDK son las siguientes:

 Proporciona una interfaz de programación (API) óptima para el desarrollo de software de reconocimiento de huella dactilar.

 Proporciona una aplicación de software de ayuda y una interfaz de usuario de rápida y fácil utilización.

 Permite un fácil desarrollo mediante funciones de registro y autentificación de huellas que operan de forma transparente para el programador.

 Funciones de identificación 1:N muy rápidas.

 Hasta 10 huellas por persona.

 Permite una fácil personalización de la interfaz de usuario minimizando el coste y tiempo empleados en el desarrollo.

 Seguridad en la utilización de la información de la huella mediante un algoritmo de encriptación de 128 bits.

 Soporta la conversión de distintos formatos de imágenes de huella (BMP, JPG, WSQ, etc.)

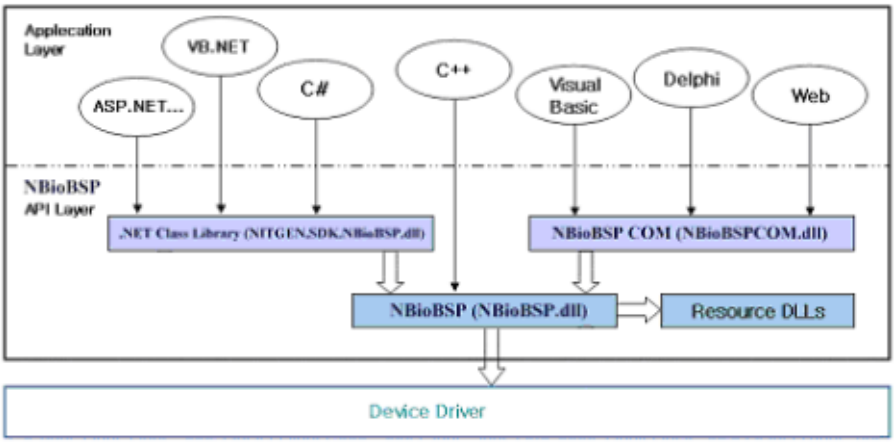


Fig. 23: Estructura eNBSP SDK [20]

4.2.3 Diseño del reclutamiento y verificación de huellas dactilares

A diferencia de la gestión de usuarios, para la cual se ha usado una arquitectura cliente servidor, el registro de huellas se hará de forma local. El propósito de la aplicación es el reclutamiento de huellas, y una vez estés registradas habrán de verificarse en dos fases; Primera y segunda visita.

Una vez se tengan usuarios en la base de datos se podrá realizar el reclutamiento de sus huellas dactilares. La función de reclutamiento recibirá el identificador único del usuario y se tomarán muestras de huellas de dos dedos, que serán ambos índices. De cada dedo se tomarán 4 muestras para posteriormente construir y grabar un patrón para cada dedo a partir de las 4 huellas tomadas de cada uno.

Cuando un usuario además de estar dado de alta haya realizado la fase de reclutamiento de huellas podrá realizar la verificación. La primera y segunda visita son procesos análogos, con la única diferencia de las muestras tomadas en un caso y en el otro. Al igual que la función de reclutamiento, las de verificación recibirán el identificador del usuario. En la primera visita se tomarán 6 muestras de cada dedo para realizar la verificación y en la segunda visita se tomarán 10 muestras. Una vez se tome una muestra, se comparará con el patrón obtenido en el reclutamiento. En el caso de que la huella no se verifique correctamente se volverá a pedir otra muestra hasta que los reintentos lleguen a 3, en cuyo caso se mostrará un mensaje de error y se volverá al principio del programa. Las huellas del reclutamiento se grabarán como un archivo binario cuyo nombre será una cadena con el siguiente formato:

HMS_00000X_0Y_ENR.FIR

- X será el número de identificación del usuario.
- Y será un dos en caso de que el dedo sea el índice derecho y un siete si es el índice izquierdo.

Fig. 24: Flujo del reclutamiento

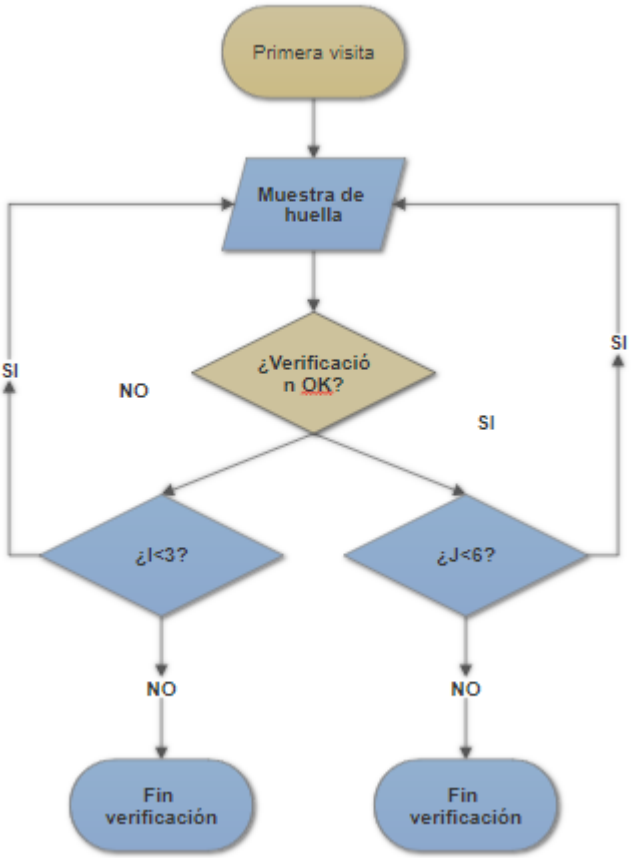


Fig. 25: Flujo de la primera visita

5 Desarrollo

5.1 Base de datos

Al comienzo del proyecto, y después de haber tenido en cuenta todas las consideraciones de diseño mencionadas en el capítulo anterior, lo primero que se comenzó a desarrollar fue la gestión de la base de datos.

5.1.1 Estructura general de la gestión de usuarios

Al iniciar, el programa nos ofrece un menú en el que debemos seleccionar si lo que deseamos es dar de alta un nuevo usuario, o si por el contrario lo que deseamos es buscar uno que ya esté dado de alta para iniciar un registro de huella dactilar. Dicho menú es el siguiente:

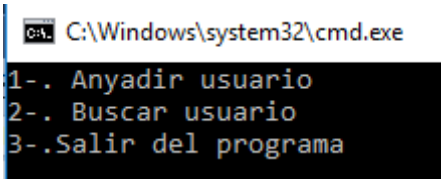


Fig. 26: Menú principal

En el caso de que se seleccione la primera opción (1-. Anyadir usuario), el programa procederá a solicitar todos los datos del usuario. Para varios de esos campos se comprobará que se han introducido correctamente, en el caso del sexo únicamente comprobando que se ha introducido un cero o un uno y en el caso del DNI y la fecha de nacimiento se llamará a sendas funciones de verificación. Una vez introducidos todos los datos correctamente, se verificará que el usuario no está dado previamente de alta en la base de datos, y si así fuera se le dará de alta. En el caso contrario se mostrará un mensaje de error y no se realizará el alta. Al finalizar ambas acciones se volverá al inicio del programa, al cual se volverá siempre. Solo se podrá salir del programa si en el menú inicial de pulsa la última opción (3-. Salir del programa).

En el caso de que en el menú inicial se seleccione la segunda opción (2-. Buscar usuario), se desplegará un nuevo menú que nos solicitará de qué forma queremos buscar un usuario dentro de la base de datos. Las opciones son DNI, email, teléfono, nombre o apellidos. Una vez elegida la opción e introducidos los datos, se nos mostrará una lista con todos los datos de todos los usuarios que cumplan los criterios de búsqueda (en el caso del DNI, email y

teléfono solo uno ya que son únicos), y se nos pedirá que mediante el identificador se seleccione al usuario que nos interese y se mostrarán todos los datos suyos únicamente. En el caso de que no hubiese ningún usuario que coincidiese con los criterios de búsqueda se sacará un mensaje de error y se volverá al principio del programa.

5.1.2 Creación de la base de datos

El primer paso de todos fue crear la base de datos y realizar sencillas pruebas con ella. Para crear la base de datos primeramente se descargó XAMPP y dentro de su control panel se activaron los módulos Apache y MySQL. Una vez iniciados ambos módulos desde la URL localhost/phpmyadmin en varios sencillos pasos se creó una base de datos llamada tfg y dentro de ella una tabla de usuarios con todas las filas necesarias para nuestra gestión de usuarios. Manualmente se introdujo un usuario. Como se ha mencionado anteriormente, al crear la tabla, se creó el campo identificador como clave primaria y los campos DNI, teléfono y email como claves únicas.

5.1.3 Interacción base de datos - aplicación C++ (Petición web)

Una vez creada la base de datos hay que poder gestionarla de una forma eficaz y segura desde nuestra aplicación C++. Una vez descartado el diseño inicial de acceso directo, surge el reto de desarrollar una arquitectura cliente- servidor para su gestión. Los pasos para conseguirlo fueron los siguientes:

- Instalar servidor
- Crear base de datos
- Conectar aplicación C++ con una página web
- Conectar aplicación C++ con script php
- Conectar aplicación C++ con script php mandando datos por POST

Después de buscar documentación sobre cómo realizar estas tareas, se decidió utilizar la librería cURL para realizar la petición web desde la aplicación C++ a los scripts php.

cURL es un proyecto de software consistente en una biblioteca (libcurl) y un intérprete de comandos (curl) orientado a la transferencia de archivos. Soporta múltiples protocolos, y entre ellos HTTP que es el necesario para nuestra aplicación. Además, cURL soporta certificados HTTP POST. El principal propósito y uso para cURL es automatizar transferencias de archivos o secuencias de operaciones no supervisadas. Es por ejemplo, una herramienta válida para simular las acciones de usuarios en un navegador web.

Al realizar las operaciones para instalar y configurar cURL en el entorno de desarrollo Visual Studio surgieron bastantes dificultades y fue complejo conseguir configurar todo correctamente para que la librería compilase. En cierto momento se barajó un cambio de sistema operativo de Windows a Ubuntu para el desarrollo del proyecto aunque finalmente se descartó. En Anexo B se explicará paso por paso la configuración final.

Estructura básica de una petición web mediante cURL enviando datos por POST:

```
curl_global_init(CURL_GLOBAL_ALL);
tamanyoDatos = strlen(datos);
    /* get a curl handle */
    curl = curl_easy_init();
    if (curl) {
        /* First set the URL that is about to receive our POST. This URL can
        just as well be a https:// URL if that is what should receive the
        data. */
        curl_easy_setopt(curl, CURLOPT_URL, "http://localhost/buscar_apellido.php");
        curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, tamanyoDatos);
        /* Now specify the POST data */
        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, datos);
        /* size of the POST data */

        curl_easy_setopt(curl, CURLOPT_WRITEDATA, response);
        int rwf = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_callback);
        if (rwf != CURLE_OK) {
            cout << "Fallo al establecer la callback";
        }

        if (rwf != CURLE_OK) {
            cout << "Fallo al establecer buffer";
        }

        /* Perform the request, res will get the return code */
        res = curl_easy_perform(curl);
        /* Check for errors */
        if (res != CURLE_OK)
            fprintf(stderr, "curl_easy_perform() failed: %s\n",
                curl_easy_strerror(res));
        curl_easy_cleanup(curl);
        curl_global_cleanup();
        char *c = response;
```

Fig. 27: Ejemplo de uso cURL

- **curl_global_init()**

Esta función configura el entorno de programación que libcurl necesita. Debe ser llamada al menos una vez dentro de un programa antes de que el programa llame a cualquier otra función de libcurl. El entorno que se establece es constante durante la vida del programa y es el mismo para cada programa, por lo que varias llamadas tienen el mismo efecto que una llamada. Los datos que se le envían son un patrón de bits que le dicen a libcurl exactamente qué características debe inicializar. En un funcionamiento normal se debe especificar CURL_GLOBAL_ALL.

- **curl_easy_init()**Esta función devuelve un identificador CURL que se debe utilizar como input para otras funciones. La llamada a esta función debe tener una llamada correspondiente a la función *curl_easy_cleanup()* cuando la operación se haya completado.

- **curl_easy_setopt()**

curl_easy_setopt() se utiliza para decirle a libcurl cómo debe comportarse. Al configurar todas las opciones, esta función hará que libcurl se comporte de una manera u otra. Todas las opciones se establecen con una opción seguida de un parámetro. Sólo puede establecer una opción en cada llamada de función. Una aplicación típica utiliza muchas llamadas *curl_easy_setopt()* en la fase de configuración.

Al pasarle el parámetro CURLOPT_URL, lo siguiente que se le debe pasar es la URL a la cual se quiere realizar la petición web. Al pasarle el parámetro CURL_POSTFIELDS, lo que se le pasará después serán los datos que quieren enviarse por POST en formato string. También se puede hacer una llamada anterior, pasando el parámetro CURL_POSTFIELDSIZE, que requerirá que después se pase el tamaño de la cadena que se va a enviar por POST.

- **curl_easy_perform()**

Se debe llamar a esta función después de haber hecho todas las llamadas a *curl_easy_init()* y a *curl_easy_setopt().* La llamada debe hacerse pasándole como parámetro lo que devuelve la llamada a la función *curl_easy_init().* Esta función realiza la petición y devuelve si la petición se ha realizado con éxito o si por el contrario se ha producido algún error

Se pueden hacer todas las llamadas que se deseen a *curl_easy_perform(),* siempre y cuando siempre se le pase el mismo parámetro, si se tiene la intención de transferir más de un archivo. Libcurl intentará reutilizar la misma conexión para las siguientes transferencias, lo que hará que las operaciones sean más rápidas y de esta forma se hará un menor uso de la CPU y de los recursos de red. Sólo hay que tener en cuenta que habrá que realizar llamadas a *curl_easy_setopt()* para establecer las opciones de la siguiente llamada a *curl_easy_perform()*.

- **curl_easy_cleanup()**

Esta función es la opuesta a la función *curl_easy_init()* y se debe llamar con el mismo identificador que devolvió la llamada a *curl_easy_init()*. Esto cerrará todas las conexiones que este identificador ha utilizado y, posiblemente, ha mantenido abiertas hasta ahora, a menos que se adjuntara a un identificador múltiple al realizar las transferencias. No se debe llamar a esta función si se tiene la intención de transferir más archivos.

- **curl_global_cleanup**

Esta función libera los recursos previamente adquiridos por la función *curl_global_init().*La función *curl_global_cleanup()* debe ser llamada una vez por cada llamada que se haya realizado a la función *curl_global_init(),* después de haber terminado de usar libcurl.

Una vez implementadas estas funciones, por la forma en la que deben mandársele los datos a la función *curl_easy_setopt()* (en formato string, solo una cadena), surgió la necesidad de tener todos los datos a enviar que en un principio estaban en distintas cadenas, en una sola

cadena. Esto se consiguió empleando las instrucciones `strcpy_s` y `strcat_s`. La primera instrucción copia una cadena en otra distinta, y la segunda concatena distintas cadenas. De este modo ya se tienen los datos listos para ser enviados por el método POST.

Una vez realizada la parte cliente en C++, se implementó la parte servidor mediante los scripts php, en los cuales se ejecutan las instrucciones propias del lenguaje SQL. Dichos scripts deben estar ubicados en un directorio ubicado dentro de la carpeta XAMPP y dentro a su vez de otra subcarpeta llamada htdocs.

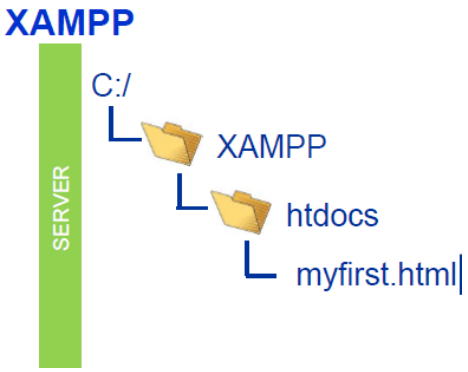


Fig. 28: Ubicación scripts php [21]

En un primer momento se consideró la implementación de un archivo php para cada distinta llamada a la base de datos, es decir, uno para la introducción de usuarios y un archivo para cada tipo de consulta. Finalmente solo se han implementados dos archivos, uno para introducir y otro para consultar, y en el archivo de consulta se ha implementado un algoritmo que identifica si lo que le está llegando es el DNI, teléfono, email, nombre o apellidos y utiliza esa información en consecuencia.

- Fichero para la introducción de usuarios

Primeramente se hacen las inicializaciones de algunas variables que serán usadas. Algunas serán para realizar la conexión a la base de datos y configurar el servidor donde está ubicada, el nombre de usuario y la contraseña, y también el nombre de la base de datos a la que queremos conectarnos. Otras son los datos recibidos mediante el método POST. Posteriormente se realiza la conexión a la base de datos, y en caso de que haya algún error en la conexión se mostrará un mensaje de error. Más tarde se ejecuta la sentencia para introducir datos en la tabla y acto seguido se realiza la query; si el usuario no está dado de alta se introduce, si no se muestra un mensaje de que ya estaba dado de alta. Finalmente se realiza la desconexión. Hay que destacar, que el control sobre si un usuario ya está dado de alta al introducirlo o no, no se hace mediante código, ni en el programa C++ ni en el script php, el control se hace directamente por la definición de los datos que se ha hecho previamente en el diseño y que se ha comentado anteriormente, esto es; DNI, email y teléfono son claves únicas en la tabla, con lo cual si se trata de dar de alta a un usuario, y al realizarse la introducción la base de datos detecta que alguno de los tres campos ya pertenece a algún usuario que está en la misma, inmediatamente dicho alta es denegado y se mostrará un mensaje notificándolo.

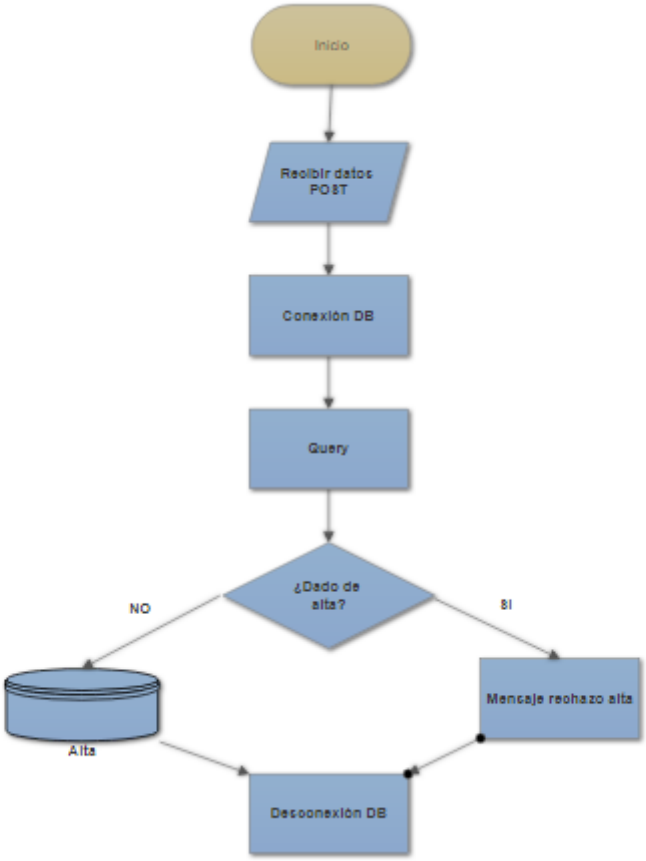


Fig. 29: Flujo alta usuarios

- Fichero para consulta de usuarios

Como se comentó anteriormente, en un principio se consideró la implementación de un fichero por cada posibilidad de consulta, pero finalmente, y por motivos de optimización de la aplicación se implementó un algoritmo que reconociese los datos que le llegan al programa, y actuase en función de ello. Como ejemplo, se mostrará un fragmento de código en el que se recibiría el DNI. Para las demás posibilidades de búsqueda el código es análogo pero modificando la variable requerida en cada caso.

```
$sql = "SELECT * FROM usuarios WHERE ";

if(array_key_exists("dni",$_POST)){
    $sql.="dni='$_POST[dni]'";
}
```

Fig. 30: Recepción condicional de datos

Como se puede observar se inicia la sentencia SQL pero se deja sin finalizar, para verificar que tipo de datos está recibiendo el programa y posteriormente concatenar la cadena para realizar la query de forma correcta. Una vez realizada la consulta, la información recibida se guarda en un array que posteriormente se codificará como un json para así enviárselo a la

aplicación cliente C++. El json enviado será una cadena con un formato que no nos sirve para las operaciones posteriores que tenemos que hacer en C++. Por ellos tenemos que darle un tratamiento a esa cadena llamado "parsear el json". Hacer un parseo significa recorrer una cadena, tenga el formato que tenga para obtener la información que es valiosa para nuestra aplicación y así poder trabajar con ella cómodamente.

Fig. 31: Flujo consulta de datos

5.1.4 Verificaciones de introducción de datos

Con el fin de proteger el programa y garantizar una correcta introducción de los datos, se han utilizado algoritmos de verificación. Para la verificación del sexo el algoritmo es trivial ya que se trata simplemente de sentencias "if" para comprobar que se haya introducido un uno o un cero, pero los algoritmos de verificación de la fecha y del DNI merecen una explicación aparte.

5.1.4.1 Algoritmo de verificación del DNI

Al ser solicitado se introduce el DNI en formato string, números y letra incluidos. Inmediatamente después de la introducción se efectúa la llamada a la función de verificación del DNI. Esta función (`bool verificar_dni(string dninumero)`) recibe como parámetro un

string que contiene el DNI que se desea verificar y devuelve un booleano, TRUE en caso de que el DNI esté introducido correctamente y FALSE si es incorrecto.

Internamente se declara un array de caracteres con tantas posiciones como posibilidades de letras hay en el DNI. Recorriendo un bucle "for" 8 veces (para leer las 8 primeras posiciones que serán los números) se suman todos los números del DNI. Si el valor ASCII de la letra introducida en el DNI, coincide con el valor ASCII de la posición resultante de obtener el resto de la suma anterior entre 23 del array declarado al principio, la función devolverá TRUE y se verificará el DNI, de lo contrario se devolverá FALSE y se deberá introducir de nuevo el DNI.

5.1.4.2 Algoritmo de verificación de la fecha

La fecha se recoge en tres variables de tipo entero, una para el día, otra para el mes y otra para el año. Una vez introducidas se llama a la función de verificación (`bool verificar_fecha(int dia, int mes, int anyo)`). Esta función al igual que antes devuelve un booleano para verificar, y recibe como parámetro los tres enteros recogidos anteriormente. Internamente lo primero que hace es comprobar que el mes sea correcto, si es correcto ejecutamos el algoritmo, si no lo es directamente se devuelve FALSE. Cuando entra al algoritmo, este se basa únicamente en un sencillo "switch" al que se le pasa como parámetro el número del mes. Al entrar en los casos del mes que corresponda, se comprueba que el día esté comprendido en 1 y 30 oentre1 y 31 según corresponda. Como caso especial, cuando la función recibe el mes 2 (Febrero) lo primero que se hace es comprobar si el año pasado es bisiesto no para determinar si los días deben estar comprendidos entre 1 y 28 o entre 1 y 29.

5.2 Reconocimiento de huella

Para empezar, se mostrará un esquema de funcionamiento típico de una aplicación de reconocimiento de huella dactilar:

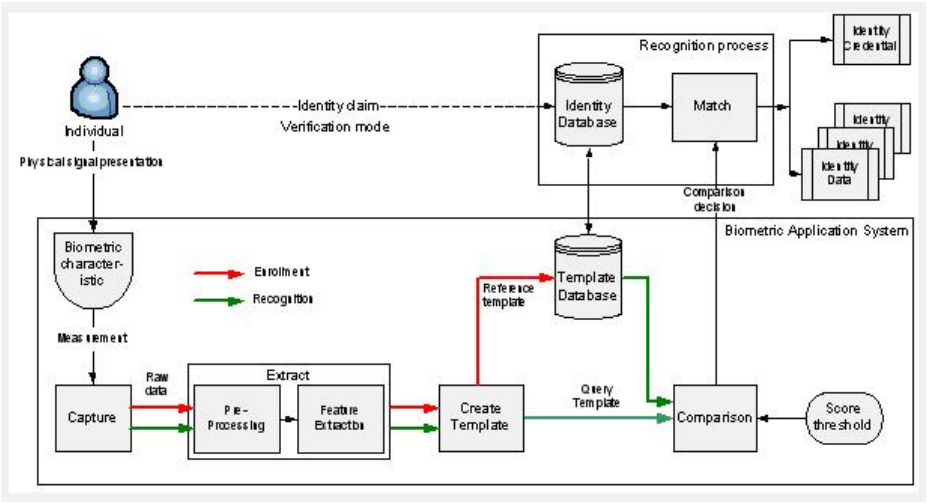


Fig. 32: Esquema funcionamiento aplicación de detección de huella [22]

Una vez terminado toda la implementación de la parte de gestión de usuarios mediante base de datos, se procedió a desarrollar el diseño propuesto para la detección de huella dactilar. Como se ha mencionado, la detección de huella está formada por tres fases:

- Reclutamiento o enroll
- Primera visita
- Segunda visita

El reclutamiento se trata del registro de huellas de un usuario una vez está dado de alta. Los procesos de primera y segunda visita son los de verificación de la huella previamente registrada en la fase de reclutamiento. Ambos procesos son análogos, con la salvedad de que en la primera visita se tomarán 6 muestras de cada dedo y en la segunda visita se tomarán 10 muestras.

Previo a comenzar el desarrollo en sí mismo, se instalaron en el PC tanto eNBSP SDK como los drivers necesarios para que el sistema reconozca el sensor de huella. Tanto para el desarrollo de la fase de reclutamiento como para las fases de primera y segunda visita, se ha seguido el manual de programación para desarrolladores C++ de eNBSP SDK.

5.2.1 FIR

Los datos de huellas dactilares procesados en NBioBSP están representados en el formato de registro de identificación de huellas dactilares (FIR en inglés). El FIR puede incluir todos los tipos de datos de huellas digitales, incluyendo imágenes en bruto, datos o datos de minucias. El FIR está formado por tres partes:

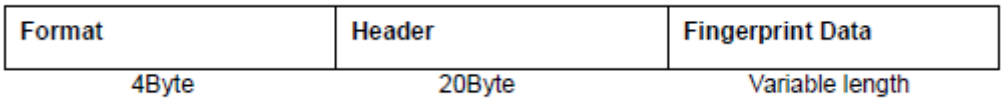


Fig. 33: FIR [23]

 Formato

Este campo indica el formato que tendrán los datos de huella dactilar. Su longitud son 4 bytes

 Cabecera

La longitud de la cabecera son 20 bytes, y se compone de los siguientes campos:

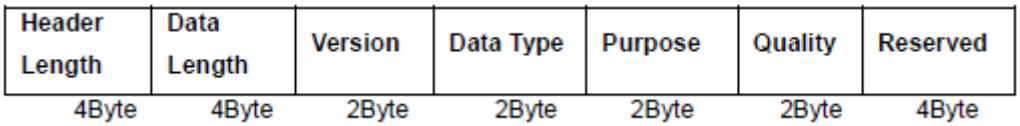


Fig. 34: Cabecera del FIR [24]

Header length y Data length contienen información acerca de la longitud de la cabecera y de los datos. El campo Version contiene información acerca de la versión del FIR. Data Type nos indica el tipo de huella dactilar almacenada en el FIR. Hay tres tipos de datos; imagen en bruto, datos intermedios y datos procesados (minucias). El campo Purpose aporta información sobre el propósito de dicho FIR (reclutamiento, verificación o identificación). Quality indica la calidad del FIR dentro de una escala que va de 0 a 100

 Datos de huella dactilar

Este campo contiene la huella dactilar en sí misma. Tiene una longitud variable y el tamaño de los datos de huella digital puede verse afectado por los valores del campo formato. El tamaño de los datos de huella digital está almacenado en el campo de longitud de datos o en la cabecera.

5.2.2 Estructura de una aplicación de reconocimiento de huella

Antes de ejecutar las funciones propias de reclutamiento y verificación, hay que realizar una serie de llamadas a distintas funciones para configurar el entorno de programación.

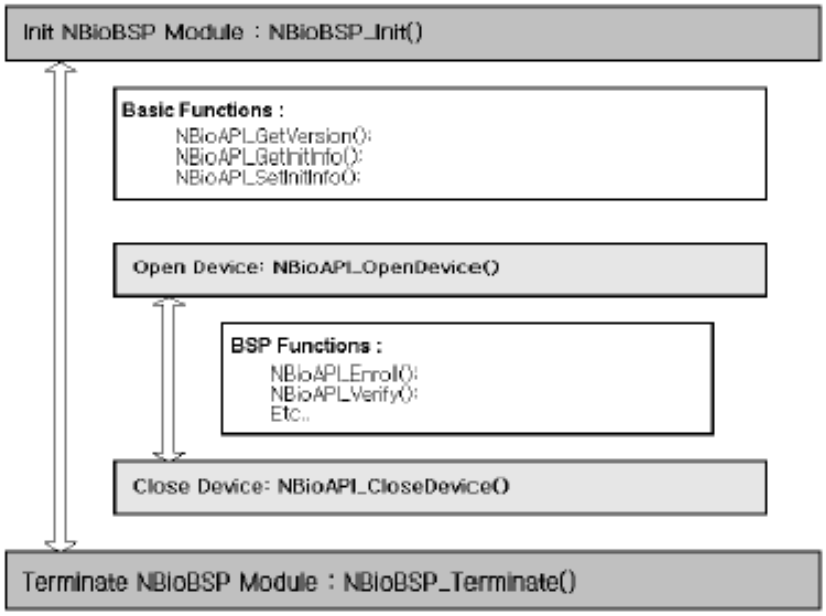


Fig. 35: esquema general de la aplicación [25]

NBioBSP.dll está firmada digitalmente para evitar manipulaciones y para garantizar la integridad de su contenido. El SDK proporciona un método para comprobar el módulo NBIOBSP.dll, y poder detectar signos de alteración. Aunque la comprobación de validez es opcional se recomienda utilizar esta función como medida de precaución. La función que se encarga de este cometido es *IsValidModule()*.Dicha función devuelve un dato de tipo bool, que será TRUE en caso de que todo sea correcto y FALSE en caso contrario.

Posteriormente, el módulo deberá ser inicializado antes de usarse. Esto se hará realizando una llamada a la función *NBioAPI_Init()*.Esta función devuelve el identificador del módulo NBioBSP. Este identificador se utilizará durante toda la duración de la sesión, es decir, durante todo el tiempo que la aplicación necesite utilizar el módulo NBioBSP. También deberá ser llamada la función *NBioAPI_Terminate()* para cerrar el módulo. Esta función libera la memoria asignada al módulo NBioBSP utilizada en la aplicación. La llamada a esta función, como es natural será hecha después de haber realizado todo el proceso de reclutamiento, verificación etc.

Después de haber usado las funciones de configuración del módulo, pasaremos a emplear las funciones del dispositivo.

La función *NBioAPI_EnumerateDeviceEx()* se usa para recuperar información específica del dispositivo; el número de dispositivos instalados y los identificadores de los dispositivos. El

identificador del dispositivo consiste en el nombre del dispositivo y su número de instancia. El byte inferior del identificador representa el nombre del dispositivo y el byte más alto representa su número de instancia. Si por ejemplo hubiese dos dispositivos FDU01 instalados en el sistema, el identificador de uno de ellos sería 0x0002 y el del otro sería 0x0102. El número máximo de dispositivos soportados es de 254.

Device Name	Value
NBioAPI_DEVICE_NAME_FDP02	0x01
NBioAPI_DEVICE_NAME_FDU01 / 04 / 06	0x02
NBioAPI_DEVICE_NAME_OSU02	0x03
NBioAPI_DEVICE_NAME_FDU11 / 14	0x04
NBioAPI_DEVICE_NAME_FSC01	0x05
NBioAPI_DEVICE_NAME_FDU03 / 13	0x06
NBioAPI_DEVICE_NAME_FDU05 / 07	0x07
NBioAPI_DEVICE_NAME_FDU08	0x08
NBioAPI_DEVICE_NAME_FDU09	0x09
(Other device)	(Refer to pDeviceInfoEx)

Fig. 36: Identificadores de los dispositivos [26]

Una vez conocido el número de dispositivos que hay instalados en el sistema, hay que inicializarlos. La función que se emplea para realizarlo es *NBioAPI_OpenDevice()*. La función comentada anteriormente nos devuelve el número de dispositivos que pueden ser inicializados, para que *NBioAPI_OpenDevice()* realice esta acción. Si no hemos realizado la llamada previamente a *NBioAPI_EnumerateDeviceEx(),* la función de inicialización inicializará automáticamente el dispositivo que esté instalado en el sistema si NBioAPI_DEVICE_ID_AUTO se especifica como el identificador del dispositivo, aunque se recomienda enumerarlos y obtenerlos identificadores. Al igual que se hizo con el módulo, también hay que cerrar el dispositivo cuando se haya terminado de usar. La llamada a la función *NBioAPI_CloseDevice()* cerrará automáticamente el dispositivo que está siendo utilizado por el módulo NBioBSP.

Queda añadir que todas las funciones nombradas anteriormente devuelven un parámetro de control para verificar si la ejecución de las mismas ha sido exitosa o si ha surgido algún fallo.

Una vez realizadas todas las operaciones necesarias tanto para configurar el módulo como los dispositivos, se realizarán las operaciones que queremos llevar a cabo en sí. El momento en que deben ser llamadas las funciones que las realizan se ve claramente en la figura 35.

5.2.3 Reclutamiento

Para el reclutamiento de usuarios se ha creado una función llamada *registrar_huella(string id)*. Esta función recibe un único parámetro, que es el número de identificación único que tiene cada usuario. Este número de identificación se usará posteriormente para asignar la huella al usuario, y será una parte del nombre del fichero que guardemos de su huella.

Lo primero que realizará esta función son las llamadas a las funciones descritas anteriormente, y cuando todo esté configurado correctamente se procederá al reclutamiento

de la huella dactilar del usuario. Como se ha nombrado al principio del apartado 5.2.1, los datos de huellas digitales procesados en el NBioBSP están en formato FIR. Los *templates* son los datos de tipo FIR utilizados para el reclutamiento. La función que realiza el reclutamiento de huella es *NBioAPI_Enroll()*. La llamada a la misma se realiza de la siguiente manera:

```
ret=NBioAPI_Enroll(
        g_hBSP,                  //Handle NBioBSP
        NULL,                    // Template guardado
        &g_hEnrolledFIR,         // Handle de FIR a ser reclutado
        NULL,                    // Payload de entrada
        -1,                      // Tiempo de espera de captura
        NULL,                    //
        NULL                     // Opciones de ventana
);
```

El FIR devuelto por la llamada a la función *NBioAPI_Enroll()* reside en el módulo NBioBSP. La aplicación no puede conocer la estructura FIR, y se refiere a ella solo como un handle (manejador).

Al llamar a la función *NBioAPI_Enroll()*, automáticamente se abre la interfaz mediante la que realizaremos el reclutamiento.

Fig. 37: Pantalla se selección de dedos

En esta pantalla se seleccionarán los dedos para los cuales se realizará el reclutamiento

Fig. 38: Pantalla de recogida de huellas

Esta pantalla nos solicita primero posicionar el dedo para la recogida de una huella, posteriormente habrá que levantar el dedo y volverlo a colocar para tomar una segunda muestra.

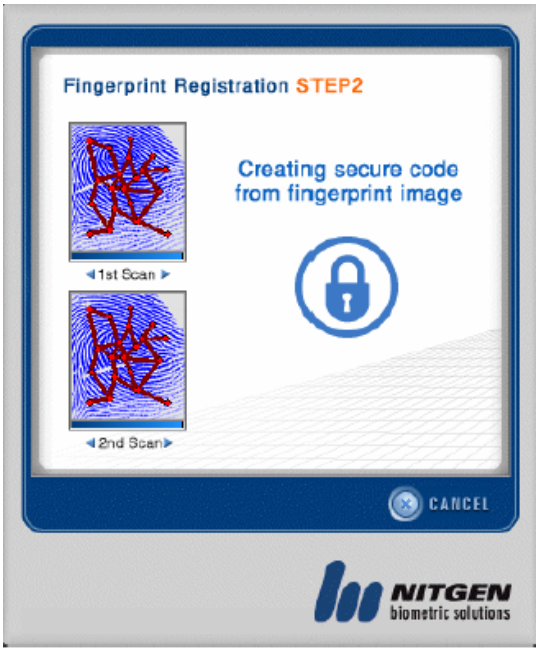


Fig. 39 Huellas recogidas

Una vez se recogen las huellas aparecerá esta última pantalla indicando que se va a generar un patrón de las huellas recogidas.

Como se ha comentado anteriormente, la aplicación no puede conocer la estructura FIR, y se refiere a ella solo como un handle (manejador), con lo cual, el siguiente paso a realizar será conseguir la estructura FIR a partir del handle obtenido con la función *NBioAPI_Enroll()*. Para ello se realizará una llamada a la función *NBioAPI_GetFIRFromHandle()*. A esta función se le pasará como parámetros el handle del módulo así como el handle obtenido en el reclutamiento. Además, se le pasará como referencia el FIR que queremos obtener. El FIR consiste en formato, encabezado y datos. La parte de datos contiene la dirección del área contigua de datos de huellas dactilares. El tamaño total del FIR es la suma del tamaño del formato, el encabezado y los datos de huella dactilar. Para posteriormente poder usar el FIR en un archivo o en una base de datos hay que convertirlo a formato "stream". Para ello se creará inicialmente una cadena dinámica de caracteres. Después se calculará la longitud del formato, de la cabecera y de los datos y se asignará a la cadena creada una longitud que sea la suma de las tres longitudes anteriores. Finalmente se hará uso de la función *memcpy()* para ir concatenando en la cadena las distintas partes del FIR. Una vez construida la cadena, esta se guardará en un archivo o base de datos y por último se liberará la memoria empleada en la cadena. Cuando el FIR y su handle no vayan a ser necesitados de nuevo, se deberá liberar la memoria que se les asignó haciendo llamadas a las funciones *NBioAPI_FreeFIRHandle()* y *NBioAPI_FreeFIR()*.

5.2.4 Verificación de la huella

Como ya se comentó, la verificación de la huella se hará en dos fases distintas, análogas la una de la otra y que solo diferirán entre sí por el número de muestras tomadas.

Al tener las huellas almacenadas en un archivo o en una base de datos, lo primero que tendremos que hacer es el proceso inverso al último que hicimos en el reclutamiento. Al igual que antes se creará una cadena dinámica. Se leerá el archivo correspondiente a la huella reclutada que se quiere verificar y se calculará la longitud del mismo para posteriormente inicializar la cadena con la longitud del archivo y rellenarla con el contenido del archivo. Al revés que se hacía antes, ahora se construye un objeto *NBioAPI_FIR* (el FIR que queremos verificar) desde la cadena, también mediante la función *memcpy()*.

Cuando tenemos construido el FIR que queremos verificar, tenemos que convertirlo a tipo *NBioAPI_INPUT_FIR*, ya que es el tipo de datos que acepta la función que vamos a emplear para realizar la verificación. Una vez obtenido este tipo de datos se hace una llamada a la función *NBioAPI_Verify()* de la siguiente forma:

```
NBioAPI_Verify(
        g_hBSP,       // Handle del módulo NBioBSP
        &inputFIR,    // FIR almacenado
        &result,      // Resultado de la verificación
        NULL,         // Payload del FIR
```

```
        10000,      // Tiempo de espera para escanear imagen
        NULL,       // Datos auditados
        NULL        // Opciones de ventana
);
```

Cuando se realice la llamada a esta función, automáticamente se abrirá la siguiente interfaz:

Fig. 40 Solicitud huella para verificación

Después de poner el dedo en el sensor:

Fig. 41 Huella para verificación capturada

Una vez capturada la huella que se quiere verificar, la función *NBioAPI_Verify()* la comparará con el FIR que hemos cargado desde el fichero. Si el resultado de la verificación es correcto, la variable result tomará el valor 1, de lo contrario tomará el valor 0.

6 Pruebas

Al tratarse de un proyecto de programación, las pruebas y la depuración de la aplicación se han ido desarrollando de forma paralela a la implementación de la misma, para así no empezar ninguna parte sin saber que la parte anterior estaba realizada correctamente y funcionando.

6.1 Base de datos

Las principales pruebas que se realizaron en la base de datos fueron las comprobaciones de que los datos introducidos por la aplicación, se almacenaban correctamente en la base de datos. La mayoría de los campos no tienen ningún control específico, pero tanto el campo DNI, como el sexo y la fecha de nacimiento si deben tenerlo.

El DNI no tiene problemas específicos al quedar grabado en la base de datos, pero si se debe verificar previamente que ha sido introducida correctamente la letra. Se realizaron pruebas con el algoritmo explicado en el capítulo de diseño y los resultados fueron exitosos.

Para la introducción del sexo, en una primera implementación se utilizó el siguiente código:

```
if (sexo) {
    strcat_s(datos, sizeof(datos), "&sexo=V");
}
else {
    strcat_s(datos, sizeof(datos), "&sexo=M");
```

Fig. 42 Primer código para el sexo

Al realizar el alta de un usuario, se observó que la base de datos quedaba de la siguiente forma:

+ Opciones

	DNI	NOMBRE	APELLIDO1	APELLIDO2	SEXO	FECHA_NAC	TELEFONO	EMAIL	ID
Editar Copiar Borrar	50559641Y	Miguel	Pintor	Montes	0	1991-08-30	628216682	miguelpintor96@gmail.com	15
Editar Copiar Borrar	42796011A	Juan	Pintor	Riquelme	0	1958-11-01	639111219	juan:pintor@terra.es	16
Editar Copiar Borrar	5879632A	Marga	Montes	Aguilera	0	1959-11-14	696301933	margamontes@gmail.com	17

Fig. 43 Error en la introducción

Como se puede observar, indistintamente de lo que se introdujese en el alta del usuario, invariablemente el sexo quedaba como un 0 en la base de datos. Para corregir este error se realizó una pequeña modificación en el código para dejarlo finalmente así:

```
if (sexo) {
    strcat_s(datos, sizeof(datos), "&sexo=1");
}
else {
    strcat_s(datos, sizeof(datos), "&sexo=0");
```

Fig. 44 Código final

Como se puede observar, la única modificación que se realizó fue sustituir la “V” por un “1”, y “M” por un “0”. De esta manera tendremos perfectamente comprobado el proceso.

Antes de implementarse la función de verificación explicada en el capítulo de desarrollo, la fecha se introducía manualmente como una cadena. El formato admitido por la base de datos para fechas es AAAA-MM-DD. En el caso de que no se introdujese correctamente la cadena en ese formato, el resultado al introducir un usuario en la base datos era el siguiente:

+ Opciones

	DNI	NOMBRE	APELLIDO1	APELLIDO2	SEXO	FECHA_NAC	TELEFONO	EMAIL	ID
Editar Copiar Borrar	50559641Y	Miguel	Pintor	Montes	1	0000-00-00	628216682	miguelpintor96@gmail.com	44

Fig. 45 Error introducción fecha

Como puede observarse, los datos introducidos son todos correctos excepto la fecha, que al ser introducida incorrectamente por teclado, en la base de datos queda reflejada como 0000-00-00. Para evitar este error, se generó un algoritmo en el que se solicitan los campos día, mes y año por separado. Primero se hace la verificación de que la fecha es correcta con la función explicada en el capítulo de desarrollo y una vez validada, se genera una cadena con día, mes y año ordenados en el formato que la base de datos espera recibir para de esta forma poder reflejarlo correctamente.

6.2 Reconocimiento de huella dactilar

Las primeras pruebas que se realizaron en la parte de reconocimiento fueron para verificar que se iniciaba correctamente el módulo, que se encontraban los dispositivos y que se iniciaban correctamente, así como que todo se cerraba de manera correcta al final. El control de todo esto se realizó mediante impresiones por pantalla de, o bien códigos que devuelven las distintas funciones que realizan estos procesos y verifican su correcto funcionamiento, o por las impresiones del dispositivo que está conectado al sistema. Todas estas pruebas dieron resultados correctos.

Una vez comprobado que todas las configuraciones previas eran correctas se procedió a desarrollar la parte de reclutamiento. Una vez desarrollada esta parte, se procedió a tomar muestras de huellas para reclutamiento y se comprobó que no se producía ningún tipo de fallo, y que además quedaban grabadas correctamente en un fichero.

Posteriormente se hizo el desarrollo de la parte de verificación. Como hemos dicho antes, la función de verificación recibe como referencia una variable llamada "result" que toma el valor 1 si la verificación es correcta y el valor 0 si es incorrecta. Al hacer la comprobación se huella con un dedo que había sido reclutado previamente, "result" tomaba siempre el valor 0 y el sistema nos indicaba que la verificación era errónea, y además en ocasiones se producían fallos bruscos en la ejecución del programa dando lugar al fin de la misma de forma imprevista. Se realizaron distintas depuraciones del programa para realizar un control sobre las variables empleadas al grabar el FIR en un fichero y también al cargarlo desde el fichero al sistema. Durante el proceso de depuración se comprobó que los procesos se estaban llevando a cabo correctamente y que los fallos en la ejecución se daban aleatoriamente en distintos puntos del código, con lo que no se pudo determinar el motivo de los errores.

El SDK ofrece otros métodos para realizar capturas y verificaciones de huellas. Esto métodos son las funciones *NBioAPI_Capture()* y *NBioAPI_VerifyMatch()*. La función *NBioAPI_Capture()* toma una sola huella, a diferencia de *NBioAPI_Enroll()* que tomaba dos y realizaba un patrón. La función *NBioAPI_VerifyMatch()* recibe dos huellas y las compara, obteniendo como parámetro también la variable "result". En un principio se realizó una prueba tomando dos huellas, e inmediatamente sin grabarlas en un fichero compararlas. Esta prueba dio un resultado positivo ya que la variable "result" tomó el valor 1, indicando que la verificación se había realizado con éxito.

Una vez comprobado que esto funcionaba, se procedió a almacenar las huellas tomadas con *NBioAPI_Capture()* en un fichero para posteriormente cargarlas en el sistema y hacer la verificación. Al realizar este proceso, se reprodujeron los mismos errores que aparecían en nuestro primer desarrollo, tanto de error de verificación como de excepciones no controladas.

Estas pruebas han permitido acotar el error. Se sabe que las funciones de reclutamiento/captura y verificación de huella dactilar funcionan correctamente, pero que existen fallos al grabar u obtener los datos desde ficheros. Estos errores se deben a fallos en la gestión de la memoria para los objetos empleados en estos procesos. Por ello, se revisó minuciosamente la parte de código referente a estos procesos. Todos los pasos seguidos han sido los expuestos por el fabricante de SDK en su manual de programación C++, con lo que podemos concluir que existen errores en dicho manual, y habrá que contactar con el fabricante para que realice una revisión del mismo.

7 Conclusiones y líneas futuras

En este último capítulo de la memoria se sacarán algunas conclusiones sobre el trabajo realizado, además de sugerir posibles ampliaciones del mismo.

7.1 Conclusiones

En líneas generales, este proyecto tenía dos principales objetivos; El primero de ellos era la gestión de usuarios a través de una base datos de una forma fiable y segura, y el segundo, la implementación de un sistema de reclutamiento y verificación de huellas dactilares.

Después de unas primeras consideraciones erróneas de diseño, se puede decir que se ha conseguido desarrollar y dotar de una funcionalidad correcta a la parte de base de datos, no así como a la parte de reconocimiento de huella dactilar. El fracaso en el segundo de los objetivos ha sido motivado presumiblemente por errores en el SDK empleado para la implementación de esta parte.

Las principales dificultades para el desarrollo de la aplicación no han venido motivadas tanto por la implementación de los algoritmos necesarios para programar la misma, sino más bien por la instalación y configuración de las librerías necesarias para poder desarrollar la programación a alto nivel. Una vez superados estos escollos se ha podido desarrollar el proyecto, hasta el momento en el que se han encontrado fallos en el SDK.

7.2 Líneas futuras

En primer lugar, se debe contactar con el fabricante del SDK para comprobar donde están los errores del manual. Una vez identificados y resueltos los mismos, habría que proceder a comprobar si las correcciones son correctas, y si lo son, realizar la parte de verificación de las huellas dactilares.

Este trabajo se ha realizado con el uso de un único sensor de huella, pero el SDK que se ha utilizado posee algoritmos para usar distintos sensores facilitados por el fabricante, con lo que se podría realizar la aplicación acoplando varios sensores en la toma de muestras, tanto ópticos como ultrasónicos.

Después de acoplar varios sensores en nuestra aplicación y comprobar que todo funciona correctamente, ya que el objeto de este proyecto era simplemente el desarrollo de una aplicación genérica que gestionase usuarios y recogiese huellas, el siguiente paso sería la implementación de una aplicación real que pudiese comercializarse y que fuese personalizable, usando la base ya obtenida en este proyecto de fondo. Una vez aprendidos los conceptos SQL y la arquitectura cliente-servidor para la manipulación de bases de datos, podrían desarrollarse bases de datos más complejas con más tablas, así como también dotar de una interfaz gráfica a la aplicación si la misma fuese necesaria para el cometido que fuese a realizar.

Bibliografía

[1] Firma digitalizada. La normalización en el campo de la biométrica. http://firmadigitalizada.net/2012/07/07/la-normalizacion-en-el-campo-de-la-identificacion-biometrica/ , Consultado: “02-09-2017”

[2] International Organization for Standarization. https://www.iso.org/standard/41447.html , Consultado “02-09-2017”

[3] Wikipedia, bases de datos. https://es.wikipedia.org/wiki/Base_de_datos , “Consultado:”02-09-2017”

[4] Maestros del web. ¿Qué son las bases de datos? http://www.maestrosdelweb.com/que-son-las-bases-de-datos/ , Consultado: “02-09-2017”

[5] CCM. Introducción a las bases de datos. http://es.ccm.net/contents/66-introduccion-a-las-bases-de-datos , Consultado: “02-09-2017”

[6] Wikipedia. MySQL. https://es.wikipedia.org/wiki/MySQL , Consultado: “03-09-2017”

[7] Isocial web. MySQL ¿Qué es y para qué sirve? https://isocialweb.agency/mysql-que-es-y-para-que-sirve/ , Consultado: “03-09-2017”

[8] Search Data Center. MySQL. http://searchdatacenter.techtarget.com/es/definicion/MySQL , Consultado: “03-09-2017”

[9] Indira informática. ¿Qué es MySQL? http://indira-informatica.blogspot.com.es/2007/09/qu-es-mysql.html , Consultado: “03-09-2017”

[10] Bibing.us. http://bibing.us.es/proyectos/abreproy/12115/fichero/Memoria%252F2_Antecedentes+y+estado+del+arte.pdf , Consultado: “03-09-2017”

[11] Wikipedia. C++. https://es.wikipedia.org/wiki/C%2B%2B , Consultado: “03-09-2017

[12] Aprendeaprogramar.com ¿Qué es php? http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=492:ique-es-php-y-ipara-que-sirve-un-potente-lenguaje-de-programacion-para-crear-paginas-web-cu00803b&catid=70&Itemid=193 , Consultado “05-09-2017”

[13] Red gráfica Latinoamérica. El lenguaje de programación php. http://redgrafica.com/El-lenguaje-de-programacion-PHP , Consultado: “05-09-2017”

[14] Monografías. Proyecto de registro y verificación biométrico en fronteras. http://www.monografias.com/trabajos102/proyecto-registro-y-verificacion-biometrico-fronteras/proyecto-registro-y-verificacion-biometrico-fronteras.shtml , Consultado: “07-09-2017”

[15] Maya Vargas, Adriana (2013). Sistema de reconocimiento de huellas dactilares en control de acceso de entrada y salida.

http://repository.unimilitar.edu.co/bitstream/10654/11168/1/MayaVargasAdriana2013.pdf , Consultado: “07-09-2017”

[16] Simón Zorita, Danilo (2003). Reconocimiento automático mediante patrones biométricos de huella dactilar. http://oa.upm.es/79/1/09200327.pdf , Consultado: “07-09-2017”

[17] Genbeta. https://www.genbeta.com/herramientas/microsoft-apuesta-por-el-multiplataforma-visual-studio-2017-llega-para-windows-macos-y-linux , Consultado: “09-09-2017”

[18] Tecnicia. http://tecnicia.blogspot.com.es/2011/03/notepad-primera-parte.html , Consultado: “09-09-2017”

[19] Digital learning. ¿Qué hace un servidor web como Apache? http://www.digitallearning.es/blog/apache-servidor-web-configuracion-apache2-conf/ , Consultado: “09-09-2017”

[20] FOC. Razones por las que usar MySQL. http://www.foc.es/2013/04/11/988-razones-por-la-que-utilizar-mysql.html , Consultado: “09-09-2017”

[21] Wikipedia. XAMPP. https://es.wikipedia.org/wiki/XAMPP , Consultado: “09-09-2017”

[22] Wikipedia. Cliente-servidor. https://es.wikipedia.org/wiki/Cliente-servidor , Consultado: “13-09-2017”

[23] SecuGen. http://www.secugen.com/download/SecuGenFingerprintReaderGuide.pdf , Consultado: “15-09-2017”

[24] NITGEN. http://www.nitgen.com/eng/product/enbsp_sdk.html , Consultado: “17-09-2017”

Anexo A: Planificación y Presupuesto

A continuación se va a llevar a cabo un desglose de las tareas que se han realizado a lo largo de este trabajo fin de grado.

A.1 Planificación

Debido a la complejidad de un trabajo de estas características se ha optado por dividirlo en distintas fases, las cuales se van a comentar a continuación:

Fase 1: Documentación inicial

En una primera fase de documentación se adquirieron conocimientos básicos en interfaces web, tanto de servidores como de bases de datos. En esta fase también se aprendieron conceptos de php para posteriormente usarlo como servidor. Para la parte desarrollada en C++ se estudió la forma de hacer peticiones HTTP mediante la librería cURL.

Una vez obtenidos los conocimientos en esas materias, el estudio se centró en los sistemas biométricos primero de forma general, y finalmente de forma más específica en los sistemas de reconocimiento de huella digital, así como un profundo estudio del manual de programación C++ del sensor utilizado para el desarrollo de la aplicación.

El estudio y documentación de conceptos generales sobre todas estas tecnologías dio lugar al estado del arte, detallado en el capítulo 2 de esta memoria.

Fase 2: Desarrollo de la aplicación

En esta fase se implementaron todos los algoritmos necesarios para la correcta gestión de la base de datos, así como para la toma de muestras de huella dactilar.

Fase 3: Pruebas

En esta fase se verificó que la base de datos gestionaba la información correctamente y se hicieron ciertas correcciones sobre el desarrollo inicial. También se obtuvieron muestras de huella y se comprobó que las verificaciones realizadas eran correctas.

Fase 4: Elaboración de la memoria

La elaboración de la memoria se ha ido realizando de manera progresiva y en paralelo a las otras tres fases durante el desarrollo del proyecto, según se iban realizando avances en cada una de las mismas. Se la considera la última fase debido a que primero se realizaban las labores de recabar información, desarrollo y pruebas, para posteriormente documentarlo en la memoria, y no se ha terminado la redacción de la misma hasta que el proyecto llegó a su fin.

FASES	HORAS EMPLEADAS
Documentación inicial	70
Desarrollo de la aplicación	130
Pruebas	10
Elaboración de la memoria	90
TOTAL	**300**

Tabla 1 – Desglose de tareas

A.2 Presupuesto del Trabajo Fin de Grado

A.2.1 Costes materiales

Los materiales necesarios han sido un ordenador con procesador Intel Core I5, en el que se ha instalado el sistema operativo Windows 10. Se recomienda que el ordenador sea de altas prestaciones para un correcto funcionamiento de la aplicación. Además, para recoger las huellas se ha hecho uso del sensor MFDU01-C3 y del software eNBSP SDK de NITGEN. Los demás elementos de software empleados han sido Open Source y por lo tanto no afectan al presupuesto. Considerando un periodo de amortización de cada uno de los dos dispositivos de 3 años y teniendo en cuenta el tiempo del proyecto, los costes materiales quedan como se expone en la Tabla 2.

CONCEPTO	PRECIO (€)
Ordenador Intel Core I5 con Windows 10	250
Sensor MFDU01-C3	25
eNBSP SDK	200
TOTAL	**475**

Tabla 2– Costes Materiales

A.2.2 Costes de personal

Para la realización de este trabajo, ha sido necesaria la presencia de un jefe de proyecto y un ingeniero.

OCUPACIÓN	HORAS	PRECIO/HORA	IMPORTE (€)
Jefe de proyecto	25	65	1625
Ingeniero	275	45	12375
TOTAL	**300**		**14000**

Tabla 3– Costes de Personal

A.2.3 Costes totales

CONCEPTO	PRECIO (€)
Costes de materiales	475
Costes de personal	14000
Costes indirectos (20%)	2895
Subtotal	17370
IVA (18%)	3126.6
TOTAL	**20496.6**

Tabla 4– Costes Totales

El coste total del proyecto es de VEINTE MIL CUATROCIENTOS NOVENTA Y SEIS EUROS CON SESENTA CÉNTIMOS.

Madrid, 20 de Septiembre de 2017

El ingeniero

Anexo B: Configuración de librerías

Durante el desarrollo del proyecto, uno de los principales problemas que ha hecho que algunas veces se encontrase en punto muerto y no se pudiese avanzar, ha sido la configuración de la librería cURL y las librerías de huella dactilar para que funcionasen correctamente en Visual Studio. Como ha sido un punto importante a la hora de desarrollar el proyecto, se ha considerado añadir un anexo con la forma en que se han configurado ambas.

B.1 Librería cURL

A continuación se detallarán los pasos realizados para la correcta configuración de cURL:

 Entrar en https://curl.haxx.se

 Download curl-7.54.0.zip

 Abrir “Símbolo del sistema de las herramientas Nativas de VS2017 x64.

 Desde el CMD, entrar en la dirección de la carpeta “...:\curl-7.54.0\winbuild”.

 Una vez dentro de la carpeta “winbuild” (desde el CMD) ejecutar:

nmake /f Makefile.vc mode=static VC=14

 A partir de este momento se genera dentro de la carpeta “curl-7.54.0” una nueva carpeta llamada “builds” donde se encontrarán 3 nuevas carpetas de nombre “libcurl-vc14-x64-reléase-static...”.

 Desde el CMD ejecutar:

nmake /f Makefile.vc mode=static VC=14 debug=yes

 Copiar la carpeta “curl”, que se encuentra en:

C:\curl-7.54.0\builds\libcurl-vc14-x64-release-static-ipv6-sspi-winssl\include\curl

Pegar dicha carpeta dentro de la carpeta del Proyecto de visual C++ donde va a ser utilizada esta librería. (Es la carpeta donde están los archivos .vcxproj).

 Copiar el archivo “libcurl.a.lib” que se encuentra en:

C:\curl-7.54.0\builds\libcurl-vc14-x64-release-static-ipv6-sspi-winssl\lib

Pegarlo dentro de la carpeta “curl” creada en el punto anterior.

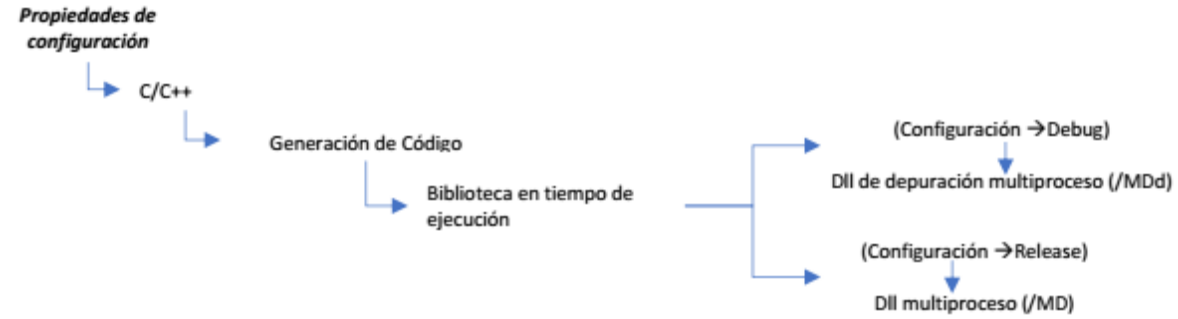
 Una vez que ha terminado la última ejecución en el CMD, se han generado también dentro de la carpeta “builds”, las carpetas “libcurl-vc14-x64-debug...”

Copiar ahora el archivo libcurl_a_debug.lib que se encuentra en:

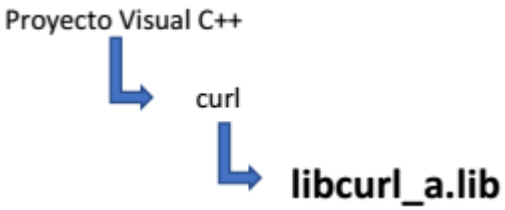
C:\curl-7.54.0\builds\libcurl-vc14-x64-debug-static-ipv6-sspi-winssl\lib

Pegarlo dentro de la carpeta “curl” pegada en puntos anteriores.

 Entrar en el proyecto de visual C++. Abrir las propiedades del proyecto y modificar:



 Desde el explorador del proyecto Visual C++, hacer click derecho sobre el nombre del proyecto y seleccionar “Agregar elemento Existente”, y seleccionar:

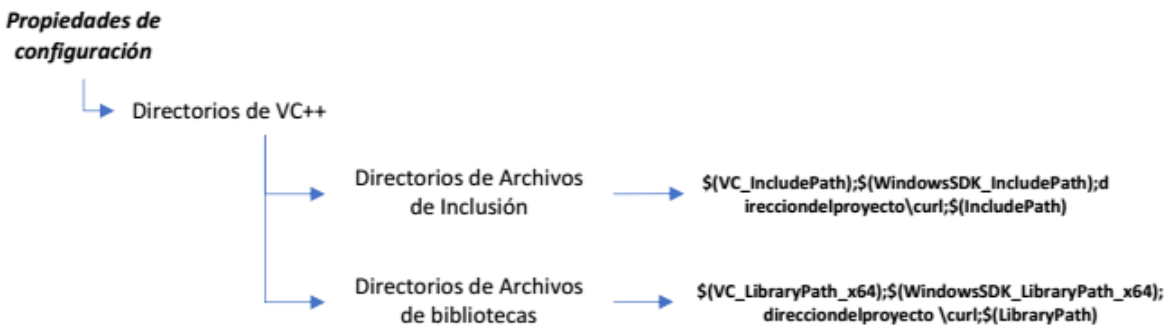


 13) Dentro del archivo principal del proyecto de Visual C++, escribir:

#define CURL_STATICLIB

#include “curl/curl.h”

 Incluir la nueva carpeta curl creada dentro del proyecto de visual C++, dentro de los directorios de VC++:



 Seleccionar la opción “Compatible con Common Language Runtime (/clr)” dentro de “General” de las Propiedades de configuración.

B.2 Librerías de huella dactilar

Ahora se detallarán los pasos para la configuración de las librerías de huella dactilar:

 Agregar directorio de archivos de inclusión (donde se encuentran los .h que vamos a importar en el código)

- Ir a C/C++ - General - Directorios de inclusión adicionales

- En este caso, para un proyecto Release de 64bits, habría que agregar la línea:
"C:\Program Files\NITGEN eNBSP x64\SDK\Inc"

 Agregar directorios de archivos de bibliotecas: (donde se encuentran los .lib correspondientes a las DLL que vamos a usar):

- Ir a Vinculador - General - Directorios de bibliotecas adicionales

- En este caso, para un proyecto Release de 64bits, habría que agregar la línea:

"C:\Program Files\NITGEN eNBSP x64\SDK\Lib\x64"

 Agregar dependencias adicionales: en este paso hay que agregar los nombres de los ficheros .lib correspondientes a las DLL que vamos a usar.

- Ir a Vinculador - Entrada - Dependencias adicionales

- Agregar, por ejemplo, NBioAPI_CheckValidity.lib y NBioBSP.lib